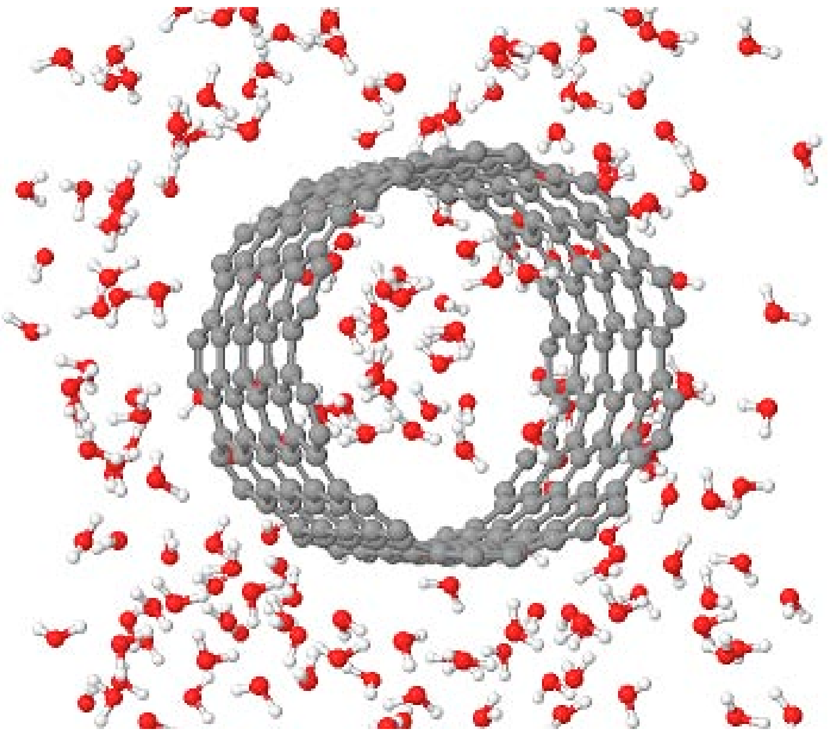
MCCCS Towhee

Monte Carlo for Complex Chemical Systems



USER MANUAL

Version 6.0.0

Contents

MCCCS Towhee

Photograph: Canyon Towhee, July 1998, Albuquerque NM

Project Administrator: Marcus G. Martin
E-mail: marcus_martin@users.sourceforge.net
Useful Bias Incorporated
88 Martinez Road
Edgewood, NM 87015

Introduction to MCCCS Towhee

Towhee Project Executive Summary

Towhee is a Monte Carlo molecular simulation code originally designed for the prediction of fluid phase equilibria using atom-based force fields and the Gibbs ensemble with particular attention paid to algorithms addressing molecule conformation sampling. The code has subsequently been extended to several ensembles, many different force fields, and solid (or at least porous) phases.

It is the hope of the developers that Towhee serves as a useful tool for the molecular simulation community and allows science to move forward more quickly by eliminating the need for individual research groups to rewrite routines that already exist and instead allows them to focus on algorithm advancement, force field development, and application to interesting systems.

A Brief History of MCCCS Towhee

The Monte Carlo for Complex Chemical Systems (MCCCS) program was first developed in 1994 in J. Ilja Siepmann's research group at the University of Minnesota starting with software dating back to the dawn of the configurational-bias Monte Carlo method. From 1994 to 1999 the code took shape as J. Ilja Siepmann, Marcus G. Martin, Bin Chen, Collin D. Wick, Jeffrey J. Potoff, and John M. Stubbs all implemented (and developed) algorithms and force fields for predicting phase coexistence using Monte Carlo. This research on MCCCS was supported by the National Science Foundation (Siepmann), and the Department of Energy (DoE) Computational Science Graduate Fellowship (Martin and Wick). In 1999 the MCCCS program was placed under a GNU general public license, but was not widely distributed.

In 2000 the DoE Office of Industrial Technologies, later renamed the Industrial Technologies Program, funded Marcus G. Martin and some industrial collaborators to make MCCCS accessible to the broad molecular simulation community by creating a web manual, improving the ease of use, and distributing the code via the web. This new version was called MCCCS Towhee, and is maintained by marcus_martin@users.sourceforge.net.

In 2003 Towhee began migrating to the SourceForge site and it finished moving there completely in September of

2005. The user base has continued to swell, and new developers are leaving their mark upon the code. Instructions for downloading, unpacking, and compiling the current Towhee release are found on the Towhee download web page.

Site Map for MCCCS Towhee Web Manual

Towhee Capabilities

Contains a list of all the Ensembles and Monte Carlo moves implemented into Towhee and summarizes all of the force fields included with the distribution.

User Manual

Contains an overview of all of the files either input or output from the Towhee code. If you aren't sure where to start, look here

Code Manual

This section explains the general structure of the Towhee program and also includes instructions on compiling Towhee and submitting bug reports.

Example Manual

This section describes the examples included with the current download version of Towhee and gives some tips for getting started with these examples.

Utility Summary

This section describes the utility programs that are included with the Towhee distribution. These utilities are designed to either make setting up simulations easier, or to provide some post processing capability.

Developer Manual

The section contains information for Towhee developers, and for those who wish to become Towhee developers.

Download Towhee

This section describes the steps required to download a version of the MCCCS Towhee program.

Monte Carlo Primer

This section provides a quick overview of statistical mechanics and discusses the reasons why somebody might want to perform a Monte Carlo molecular simulation.

References

This section contains references to papers and books that are either the source materials for the methods or force fields in Towhee, or are helpful reviews of those subjects.

Towhee Publications

A list of user publications that have utilized the Towhee code.

Towhee Presentations

A list of user presentations that utilized the Towhee code.

MCCCS Towhee on SourceForge

The SourceForge home for Towhee. This is the place to come if you wish to submit bugs, request new features, or discuss general topics related to Towhee on the forums. It also contains the CVS repository for Towhee and the Towhee Bug Tracker utility.

Related Software

A list of software packages that are useful in combination with Towhee and also a list of other publicly available Monte Carlo molecular simulation packages.

Send comments to: Marcus G. Martin

Last updated: March 1, 2007

GNU General Public License

Table of Contents

GNU GENERAL PUBLIC LICENSE

Version 2, June 1991

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

- **a)** You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.
- **b)** You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.
- **c)** If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
one line to give the program's name and an idea of what it does.
Copyright (C) yyyy  name of author

This program is free software; you can redistribute it and/or
modify it under the terms of the GNU General Public License
as published by the Free Software Foundation; either version 2
of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program; if not, write to the Free Software
Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA  02111-1307, USA.
```

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

```
Gnomovision version 69, Copyright (C) year name of author
Gnomovision comes with ABSOLUTELY NO WARRANTY; for details
type `show w'.  This is free software, and you are welcome
to redistribute it under certain conditions; type `show c'
for details.
```

The hypothetical commands `` `show w' `` and `` `show c' `` should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than `` `show w' `` and `` `show c' ``; they could even be mouse-clicks or menu items--whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

```
Yoyodyne, Inc., hereby disclaims all copyright
interest in the program `Gnomovision'
(which makes passes at compilers) written
by James Hacker.

signature of Ty Coon, 1 April 1989
Ty Coon, President of Vice
```

MCCCS Towhee (Capabilities)

Summary

Towhee is a Monte Carlo molecular simulation code that was designed for computing phase equilibria, but is suited for other tasks as well. This documentation contains a list of the current capabilities of the Towhee code.

Ensembles

- Canonical (NVT)
- Isobaric-isothermal (NpT)
- Grand Canonical (uVT)
- Gibbs (both NVT and NpT)

Force Fields

12-6 plus 12-10 H-bond
- Weiner *et al.* 1984
- Weiner *et al.* 1986
- Dick and Ritchie 1994

12-6 plus solvation
- Charmm19 EEF1
- Charmm19 SASA

12-9-6
- Frischknecht and Curro 2003

9-6
- COMPASS Version 1
- QMFF-VIII

Buffered 14-7
- MMFF94

Double Exponential
- MCY 1976 water

Embedded Atom Method
- Ackland *et al.* 2004 Fe and P
- Hoyt *et al.* 2003 Cu and Pb
- Mendelev *et al* 2003 Fe

Exponential-12-6
- Catlow and Faux

Exponential-6
- Kramer Farragher van Beest van Santen (also known as BKS)
- MM2

Gordon n-6
- Gordon

Hard 2580 Multistep or Repulsive 2580 Multistep
- Cui and Elliott 2002

Hard Sphere or Repulsive Sphere
- Hard Sphere

Lennard-Jones
- Alavi *et al.* H_2
- Amber param96
- Aqvist 1990 cations
- Charmm22
- Charmm22fe
  supplementary parameters for fluoroethanes in CHARMM22
- Charmm27

- Charmm27x some supplementary parameters for CHARMM27
- Charmm27 rigid water
- ClayFF
- Coon *et al.* 1987 O_2 and N_2
- Cui *et al.* 1998 perfluorinated alkane models
- DACNIS United Atom
- DREIDING
- Dubbeldam *et al.* alkanes and zeolites
- EPM family of CO_2 models
- Galassi and Tildesly 1994 diatomic gasses
- Gromos 43A1
- Jaramillo *et al.* 2001 Hydrofluorocarbons
- Lastoskie *et al.* 1993 N_2
- Lennard-Jones beads
- Lybrand Ghosh McCammon 1985 anions
- Morrow and Maginn 2002 ionic liquids
- NERD United Atom (Version 1)
- NERD United Atom (Version 2)
- NERD United Atom (Version 3)
- OPLS-aa
- OPLS-ua
- OPLS-1996
- OPLS-2001
- Panagiotopoulos 1989 noble gasses
- Potter *et al.* 1997 fluoromethanes
- Richards *et al.* 1995 N_2 and O_2 in zeolite Li-X
- Shah and Maginn 2004 ionic liquids
- Shukla 1987 gasses
- SKS n-alkanes
- Smith and Dang 1994 NaCl
- SMMK (main text) alkanes
- SMMK (note added in proof) alkanes
- SPC-E water
- Sum *et al.* 2003
- Teleman *et al.* 1987 water
- TIP3P water
- TIP4P water
- TIP5P water
- TraPPE Explicit Hydrogen
- TraPPE United Atom
- TraPPE United Atom flexible bonds
- Walther *et al.* 2001 carbon nanotubes and water

Multiwell
- Elliott 2002 *n*-alkanes and benzene

Stillinger-Weber
- Ding and Anderson 1986 Ge
- Stillinger and Weber 1985 Si
- Vink *et al.* 2001 Si

Square Well or Repulsive Well
- SquareWell
- Vega *et al.* 1992

Tabulated Pair
- No forcefield files currently available for this potential.

UFF 12-6

    - Universal Force Field (UFF)
- Other (works with multiple classical_potential values)
    - MGM Stereochem enforcer

Monte Carlo Moves

- Isotropic volume change by scaling the center-of-mass of all molecules in the simulation box (McDonald 1972)
- Non-isotropic adjustment of the unit cell vectors.
- Rotational-bias interbox molecule transfer (Cracknell *et al.* 1990)
- Coupled-decoupled configurational-bias interbox molecule transfer
- Coupled-decoupled configurational-bias intrabox molecule transfer
- Coupled-decoupled configurational-bias molecule creation or deletion (grand canonical)
- Aggregation-volume-bias move number 1 (Chen and Siepmann 2000)
- Aggregation-volume-bias move number 2 (Chen and Siepmann 2001)
- Aggregation-volume-bias move number 3 (Chen and Siepmann 2001)
- Coupled-decoupled configurational-bias regrowth for linear, branched and cyclic molecules
- Fixed endpoint Coupled-decoupled configurational-bias regrowth of interior segments along a polypeptide backbone.
- Pivot of one portion of a molecule about a torsional axis. (first used by Lal 1969)
- Concerted rotation of a portion of the molecule backbone (Dodd *et al.* 1993)
- Translations of an entire slab of molecules
- Translations of an entire cylinder of molecules
- Translation of a single atom in a molecule
- Translation of an entire molecule (Metropolis *et al.* 1953)
- Rotation about the center-of-mass

Return to the main towhee web page

Send comments to: Marcus G. Martin

Last updated: January 05, 2007

MCCCS Towhee (User Manual)

Overview

This section covers the basic information that is needed to set up the input files for Towhee and to get some useful work done with the code. It also describes the files that are generated by the Towhee program. Note that Towhee has a version control number that is related to the input files. The first number changes only when modifications to Towhee require a new format for towhee_initial, the second number changes only modifications to Towhee require a new format for towhee_input, and the final number is changed when Towhee is modified in such a way that none of the input files have been altered (generally for bug fixes). For information about the Towhee program files see the Towhee code manual.

Several illustrative test cases are included with the Towhee download. Information about running those test cases can be found in the Towhee example manual.

Towhee Input Files

towhee_input (input file)

This is the main input file for Towhee and is generally the only file that needs to be edited on a regular basis. It has a regimented style to the input. The variables are described here in the order they appear in this file. Please look at one of the example files (included with the code release) for the precise file format.
Current Release version
towhee_input Version 6.0.x
Older input file versions
towhee_input Version 5.3.x
towhee_input Version 5.2.x
towhee_input Version 5.1.x
towhee_input Version 5.0.x
towhee_input Versions 1 through 4

towhee_coords (input file)

This file is used when you are first setting up the run and you want the code to directly read the initial coordinates (**initstyle**='coords'). The code will read one set of atom coordinates from each line (x,y,z). Note that the code builds the entire first box configuration before moving on to subsequent boxes and it also places all molecules of type 1 in the box before placing those of subsequent types. This file is most useful when you already have a coordinate file from another source (such as a pdb file, or output from another code) and just want to directly read that into Towhee.

A **towhee_coords** file may also be created from a **towhee_final** file (see below) by stripping all data preceding the atom coordinates. This technique can be useful for restarting from final coordinates while allowing **hmatrix** and maximum displacement parameters to be defined in the **towhee_input** file.

towhee_ff (input file)

This file contains all of the force field parameters for any user-defined force field. Several common forcefields are provided with the download package. This file follows a very rigid format which is described in the Towhee Force Field File documentation.

towhee_initial (input file)

This file contains the box lengths, maximum displacements, and coordinates for the system. It is used when you wish to restart a job from the final configuration of a previous run. The previous run output (either towhee_final or towhee_backup) is simply copied to towhee_initial and the next simulation continues from this configuration. Note that you must set **linit** to .false. when you want to restart and that there is some error checking to make sure the information in towhee_initial matches information in towhee_input. The code can read in previous versions of the towhee_initial file without difficulty as the version number is at the top of the file. It is possible to directly edit this file, but if you are clever enough to play around in towhee_initial then you are smart enough to look in the code (rwconf.F) to figure out how to do it. Please note that when **towhee_initial** is used, the **hmatrix** parameter, as well as parameters describing maximum move sizes, in **towhee_input** are ignored.

towhee_parallel (input file)

This file specifies the style of parallelism to use when running the parallel version of the code. This file is not needed for a serial job run using the towhee version of the code. Only the current release version is maintained here.

Current Release version

towhee_parallel

towhee_partial (input file)

This file is used when you are first setting up the run and you want the code to read in a partial list of atom positions, match them against the expected positions (specified in towhee_input), and then use configurational-bias to fill in any missing atoms. It is suggested that you generate this file using the pdb2towhee.F utility. If you are using this file to generate multiple molecules in a single run then you can simply cat all of the files together and then remove all of the "nul" lines, except for the final instance.

towhee_template (input file)

This file is used to generate an initial template for molecules when first setting up the initial configuration (**linit**=.true. and **initstyle**='template'). The molecules are read from the template in the order they are listed in towhee_input and note that if any of the boxes are set to **initstyle**='full cbmc' for one of the molecules then that molecule template is generated via configurational-bias in all of the boxes and towhee_template is not used. The format for towhee_template is one unread line (where you can put in a note to remind yourself the name of the molecule for that template) and then the x,y,z coordinates for each atom with one atom per line. This file works best if the first atom is placed at (0.0,0.0,0.0).

Towhee Output Files

standard output (output file)

Towhee outputs quite a bit of run time information to standard output (Fortran unit 6) and normally you will want to redirect this output to a filename of your choosing. That is done using standard Unix commands: for example to redirect the standard output to a file named "towhee_output" you would execute the following command.

```
./towhee > towhee_output
```

For more information about this output file, and a discussion of the chemical potential and pressure calculations, please see the Towhee standard output manual.

box_xx_step_nnn.pdb (output file)

This file is output for each box (where xx is replaced by the box number and nnn by the step number) at the beginning, end, and intermediate points of a simulation, as specified by the **pdb_output_freq** variable in the towhee_input file. Rasmol is a freely available program for visualizing pdb files.

towhee_altinp (output file)

This file contains all of the information needed to create the molecule input file for **inpstyle**=0 needed in towhee_input. This is output whenever you are using an **inpstyle** other than 0 in towhee_input. For very large molecules (like proteins) it can take a considerable amount of time to setup the force field (on the order of 20 minutes) so there are occasions when you only want to do this the first time you run and then use the **inpstyle**=0 for subsequent runs. If you want to use the **inpstyle**=0 information then just paste this file into the appropriate section of towhee_input. This file is also useful for seeing exactly which force field interactions are used for every term in case you want to check the builder to make sure it was doing what you thought it should be doing.

towhee_backup (output file)

This file contains the box lengths, maximum displacements, and coordinates for the system. It is output periodically from the code (see **backupfreq** in the towhee_input section) and can be used to restart the simulation from this configuration. In order to restart using this configuration you need to copy towhee_backup to towhee_initial (see towhee_initial).

towhee_restart_nnn (output file)

This restart file is identical in format to **towhee_backup**; the nnn suffix, which corresponds to the step at which it was written, makes the filename unique for a given run, so that new towhee_restart files do not overwrite old ones. This is useful if you wish to restart from intermediate points of a simulation. towhee_restart is controlled by the **restartfreq** parameter in the towhee_input.

towhee_final (output file)

This file contains the box lengths, maximum displacements, and coordinates for the system. It is output at the end of the simulation and can be used to restart the simulation from this configuration. In order to restart using this configuration you need to copy towhee_final to towhee_initial (see towhee_initial).

towhee_histogram (output files)

This file contains run information output during a grand-canonical ensemble run and are suitable for use in the Histogram Reweighting method. The utility program analyse_histogram is used to process this output file.

towhee_movie (output file)

This file contains a series of snapshots of the positions of all of the atoms in each simulation box. The frequency of this output is controlled by the **moviefreq** variable (set in towhee_input). This movie file allows the user to compute quantities that depend upon positions as a post-processing operation. The utility program analyse_movie computes many different distribution functions from towhee_movie.

towhee_ostwald (output file)

This file contains information that is useful when computing partition coefficients and free energies of transfer. The Ostwald coefficient (the ratio of densities in two phases/boxes) is computed and used to compute the free energy of transfer. The output contains the box numbers, partition coefficients, standard deviations of partition coefficients, free energies of transfer, and standard deviations of free energies of transfer for each molecule type in the simulation. Partition coefficients are not output if the densities are essentially zero in the second box (to avoid dividing by zero).

towhee_vlcc (output file)

This file is only generated if you have at least 1 block in the block averages, there are 2 simulation boxes (**numboxes**=2), and there is only one molecule type (**nmolty**=1). It contains a single line of information with the temperature, the gas phase specific density, the standard deviation of gas phase specific density, the liquid phase specific density, and the standard deviation of the liquid phase specific density. This is useful when constructing vapor-liquid coexistence curves. The box with the higher density is assumed to be the liquid phase, and the other box is the vapor phase. See the fitcoex utility routine for more information about the uses of the towhee_vlcc files.

Database Input Files

database_input (input file)

This file is only used when the **inputformat** is set to 'Database' and is designed for use in fitting force fields. See the database_input manual for more information.

LAMMPS Input Files

lammps_data (input file)

This file is only used when the **inputformat** is set to 'LAMMPS'. It is read into the code and used to create an initial conformation and force field file. This exists to facilitate transfer of simulation data between Towhee and the LAMMPS massively parallel molecular dynamics code.

Return to the main towhee web page

Send comments to: Marcus G. Martin

Last updated: August 9, 2008

MCCCS Towhee (code manual)

Overview

This section explains the general structure of the Towhee program and also includes instructions on compiling Towhee. This page is only maintained for the current version of the Towhee code. It was last updated for version 5.0.1.

Compiling

Towhee versions 3.9.8 or later have a sophisticated configure and Makefile setup that should allow automatic setting of the Makefile parameters on most platforms. This requires the use of two fairly standard freely available programs.
autoconf Version 2.59 or later and
automake Version 1.7.4 or later

Compiling on a Unix/Linux Operating System using compilers similar to GNU

In the top level directory issue the command
./configure
This should automatically set up Makefiles in the Source directory which will create an optimal build for your machine. Then go into the Source directory
cd Source
and type
make towhee

Compiling on an Apple OS-X Operating System

Useful instructions and software for high performance computing on OS-X is available at http://hpc.sourceforge.net/.
Compiling Towhee on OS-X is very similar to compiling on a Unix/Linux operating system. Users on older versions of OS-X may experience something similar to the following error message when they compile.

```
ld: Undefined symbols:
restFP
saveFP
make: *** [towhee] Error 1
```

If this happens then you need to configure including the following option
./configure LDFLAGS=-lcc_dynamic
and recompile.

Compiling on a Windows Operating System

Towhee is designed for compilation in a Unix style environment. In order to compile on a Windows machine we suggest the freely available Cygwin software. Once you have installed the Cygwin environment, and the appropriate additional software (Fortran compiler, C compiler, autoconf and automake) you then follow the normal Towhee compilation procedure with one important exception. Cygwin appends a '.exe' to any executable so the commands for compilation include an extra '.exe' compared with the Unix/Linux environment.
./configure
cd Source
make towhee.exe

Configure options in Towhee

- **./configure --enable-safe-compare**
  Using this flag turns on some slightly more expensive comparison routines that serve to make the results for the Examples exactly match between different machines. It also sets bounds checking and turns on compiler warning messages. This option is used to create the answer_current and par_answer_current files in the Examples directory.
- **./configure --enable-mpi**
  This option builds an MPI version of Towhee. Note that for some machines it is actually easier to get the

machine to compile using the MPI options and you can still run single processor jobs with this version. Consult the towhee_parallel manual for more information about this option.

- **./configure --enable-internal**
  Some compilers do not include the error function intrinsics. Towhee includes some internal versions of the error function routines for use when the intrinsics are lacking and this option forces Towhee to use the internally provided functions instead of the intrinsic versions.
- **./configure FFLAGS='-fbounds-check'**
  where the items between the single quotes are the flags used for the Fortran compilation.
- **./configure F77='pgf77' CC='pgcc'**
  Sets the Fortran and C compilers to user specified values. This example shows the command needed to set the compilers to PGI pgf77 and pgcc
- **./configure ADDLIB='-L/usr/whoever/lib/'**
  Adds user specified libraries into the link command.
- **./configure --enable-tramonto**
  Links the Tramonto classical density functional code into Towhee for use as a really fancy implicit solvent. This option requires access to the Tramonto code compiled as a library. See the Tramonto web site for more information.
- **./configure --enable-lcao**
  Links the Quest quantum density functional code into Towhee for use as a really expensive force field. This option requires access to the Quest code compiled as a library. See the Quest web site for more information.
- **./configure --disable-command-line-args**
  Removes the ability of the code to read command line arguments. This also defaults to a parallel style of "jobfarm" when combined with an --enable-mpi configure option. This option exists for users whose machines are unable to properly handle command line arguments, especially for certain parallel machines.
- **./configure FFLAGS=''**
  This turns off all of the Fortran compiler flags. This is useful when compiling the forcefield utility as some of the subroutines for assigning force field parameters are so large that optmization can cause some machines to run out of memory.

Compiling the lcao version on the srnsquall machine

The srnsquall machine at Sandia was the main development platform for the combination of Towhee with the Quest quantum DFT code. The procedure for compiling the combined version has been simplified.

- **./configure --enable-internal --enable-mpi --enable-lcao --enable-link-srnsquall ADDLIB=-L/questpath**
  where **questpath** is replaced with the directory path that contains the liblcao.a library. You also need to edit the Source/Makefile produced by this configure to remove two spurious ' symbols that appears in the link line.

Compiling the lcao version on the Spirit machine

These are the directions for compiling the Towhee-LCAO version on the Sandia computer Spirit.

- **./configure --enable-mpi --enable-lcao --enable-link-spirit ADDLIB=-L/questpath**
  where **questpath** is replaced with the directory path that contains the liblcao.a library. An lcao library is in the publicly readable /home/marmart/Lib directory on that machine so using that as the **questpath** should work properly.

Compiling the forcefield utlitity

Please note that the vast majority of users only need to compile the towhee portion of the code and do not need the forcefield utility as that is only required if you wish to generate the towhee_ff files. As the towhee_ff files come packaged with the download, and the forcefield utility contains routines that are large enough to give some compilers trouble, you should not attempt to compile that package unless you know why you want it.
cd Source
make forcefield

The .h files

- **preproc.h**
  The preproc.h file is the only program file that a casual user will ever need to modify. This file contains parameters which control the dimensions of many of the most important arrays in Towhee. Depending on your preproc.h settings, and the simulation you are trying to perform, the code may produce an error message informing you that some variable exceeds some other variable. If that is the case then you will need to modify the appropriate variable in preproc.h in order to accommodate your system. You also might need to decrease some of the variables in preproc.h if your machine does not have enough memory for the arrays. Finally, some of the variables in preproc.h control certain settings that almost never change, but may be useful for debugging purposes.
  - **MAXBOX** the maximum number of simulation boxes in the system. Minimum value of MAXBOX is 1. No Maximum value for MAXBOX, but if it is set to more than 10 boxes then the pdb output will no longer function properly.
  - **MAXCUBE** the maximum number of cubelets that are used for energy biasing. Minimum value is 1. No Maximum value.
  - **NMAX** the maximum number of molecules in the system. Minimum value is 1 plus the number of molecules in the system as the extra slot is used in some of the configurational-bias moves. No maximum value.
  - **NUMAX** the maximum number of atoms in any molecule. Minimum value is 1. No maximum value.
  - **MAXATOMS** the maximum number of atoms in the entire simulation. A value that would always work is NMAX*NUMAX, but the default is set as an adjustable parameter for cases where the user has an extremely asymmetric system (for example: a protein in water) and is running out of memory. Minimum value is NMAX. No maximum value, but it is a waste of memory to set MAXATOMS above NMAX*NUMAX.
  - **NTMAX** the maximum number of types of molecules in the simulation. Minimum value is 1. No maximum value.
  - **NCHNB_MAX** the maximum number of trial sites for the nonbonded selection in a configurational-bias move. Minimum value is 1. No maximum value.
  - **NCHPRENB_MAX** the maximum number of trial sites for the pre-nonbond selection in a configurational-bias move. Minimum value is 1. No maximum value.
  - **NCHTOR_MAX** the maximum number of trial sites for the torsion angle part of a configurational-bias move. Minimum value is 1. No maximum value.
  - **NCHBN_MAX** the maximum number of trial sites for the bending angle part of a configurational-bias move. Minimum value is 1. No maximum value.
  - **NCHVIB_MAX** the maximum number of trial sites for the vibration part of a configurational-bias move. Minimum value is 1. No maximum value.
  - **MAXKMAX** the maximum value for **kmax** in the Ewald sum. Minimum value is 1. No maximum value.
  - **VECTORMAX** the maximum number of reciprocal space vectors in the Ewald sum. Starting with version 3.17.0 this variable is set based upon the value of **MAXKMAX**.
  - **MAXBLOCK** the maximum number of blocks for the block averaging. Minimum value is 1. No maximum value.
  - **FLDMAX** the maximum number of external fields in the system. Minimum value is 1. No maximum value.
  - **NNTYPE** the maximum number of different types of atoms allowed in the simulation.
  - **CROSSTYPEMAX** the maximum number of cross types, currently set to depend upon NNTYPE.
  - **MAXTABTYPE** the maximum number of types of atoms for tabular potentials. Minimum value is 1. Maximum value is NNTYPE.
  - **MAXTABLE** the maximum number of entries to describe a tabular potential. Minimum value is 1. Maximum value is NNTYPE.
  - **TVIBMAX** the maximum number of different types of vibrations in the simulation.
  - **TBENMAX** the maximum number of different types of bending angles in the simulation.
  - **TTORMAX** the maximum number of different types of regular torsions in the simulation.
  - **TIMPMAX** the maximum number of different types of improper torsions in the simulation.
  - **TAAMAX** the maximum number of different types of angle-angle terms in the simulation.
  - **TOFMAX** the maximum number of different types of special one-five terms in the simulation.
  - **THBONDMAX** the maximum number of different types of special hydrogen bond force field terms in the simulation.

    - **TBIMAX** the maximum number of different types of bond increment terms in the simulation.
    - **NNBOND** the maximum number of bonds from any atom. Minimum value is 1. No maximum value, but this has not been tested for a value above 4.
    - **SMALLEST** a value close to the smallest double precision number a computer can handle. This is usually set to 1d-300. There is no reason to adjust this unless your machine can handle significantly smaller numbers.
    - **MAXPBOX** maximum number of simulation box combinations for the volume moves. Currently set as a function of MAXBOX.
    - **MAXTOR** maximum number torsions originating from any single atom. Currently set as a function of NNBOND.
    - **MAXBEND** maximum number angles originating from any single atom. Currently set as a function of NNBOND.
    - **MAXAA** maximum number of angle-angle terms originating from any single atom. Currently set as a function of NNBOND.
    - **MAXIMPROP** maximum number of improper torsion terms originating from any single atom. Currently set as a function of NNBOND.
    - **MAXOF** maximum number of special one-five terms originating from any single atom. Currently set as a function of NNBOND.
    - **MAXDIRLENGTH** maximum number of characters in the directory length. This is used with some of the parallel features.
    - **MAXIMPLICITTYPE** maximum number of types of atoms for the implicit force fields.
    - **MAXPIVOTBIN** maximum number of bins for the pivot bookkeeping.
    - **NDUMPHIST** maximum allowable number of histogram computations before an output to file.
    - **MAXBAPROP** maximum number of properties to average using the block averages routine.
    - **MAX_FOREIGN_LAMBDA** maximum number of (lambda_lj,lambda_c) pairs for which energy evaluations will take place. Relevant only for Scaled Lennard-Jones classical potential.
    - **Pound define aliases** the remainder of this file contains a series of variables that are pound defined to be integers. This allows us to use sensible variables to describe things like classical potential styles, but still have the computational expediency of integer compares versus the relative expense of character string compares.
- **globalf.h**
  contains all of the global variables (common blocks) for the Fortran portion of the code.
- **globalc.h**
  contains all of the global variables for the C portion of the code.

The .F files

The bulk of Towhee is written in a slightly enhanced FORTRAN 77. All of the FORTRAN subroutines and functions end with .F and they also contain preprocessor directives. All modern FORTRAN 77 compilers that I have tried are capable of handling the slight enhancements to standard FORTRAN 77 that are used in Towhee. Below I list all of the .F files that are in Towhee. These are broken up into several different classes of subroutines/functions in order to make it easier to follow the logic of the code.

Main program

- **mainloop.F**: this subroutine is the main control of the towhee code. It calls readinput to read in the input files. It initializes many of the statistics and safety features, and it determines which move to perform each step, and then calls the appropriate move subroutine. It also prints out most of the measured quantities at the end of the simulation.

Input and Initialization

- **readinput.F**: this subroutine opens the towhee_input and towhee_altinp files. It then reads in the **inputformat** variable from towhee_input. This variable determines whether to call readtowhee, readlammps, or readdatabase.
- **readtowhee.F**: this subroutine reads in the data from towhee_input and towhee_initial. It calls the other input routines if they are needed, it calls the force field routines to initialize parameters, and it performs some error checking. If the code stops before performing any Monte Carlo moves, then readinput is the first place to check.

- **rwcbmc.F**: this subroutine reads in from towhee_input or write out to towhee_altinp all of the variables related to the configurational-bias Monte Carlo move settings.
- **readlammps.F**: this subroutine reads information from the lammps_input and lammps_data files in order to create towhee files for use in a Towhee simulation run. This subroutine outputs lammps_coords_## files, a towhee_altinp file, and a towhee_ff file.
- **rwforcefield.F**: this subroutine either reads or writes all of the force field information to towhee_ff.
- **readclassical.F**: this subroutine reads all of the variables that are required when **potentialstyle** is 'classical'
- **readquantum.F**: this subroutine reads all of the variables that are required when **potentialstyle** is 'quantum'
- **setpotentiallogic.F**: this subroutine sets up all of logicals that determine which additional terms are read or written for each kind of **classical_potential**
- **buildhelix.F**: this subroutine constructs a helical initial configuration using the helix_keyatom and then uses configurational-bias to grow in the rest of the atoms.
- **helixpos.F**: this subroutine returns a position on a helix.
- **buildmolec.F**: this subroutine constructs the molecule information arrays for any molecule that has an inpstyle of 2. It reads in all of the data for those molecules, and then calls assemble to put everything together.
- **buildna.F**: this subroutine constructs the molecules information arrays for any molecule that has an inpstyle of 3. It read in all of the data for those molecules, constructs a map of atom names and vibrations using the various polyxxx.F routines, and then calls assemble to put everything together.
- **buildnanotube.F**: this subroutine constructs an initial nanotube conformation and also assigns all of the bonded terms needed to construct a nanotube molecule. This is called when using inpstyle 4.
- **buildprot.F**: this subroutine constructs the molecules information arrays for any molecule that has an inpstyle of 1. It read in all of the data for those molecules, constructs a map of atom names and vibrations using the various polyxxx.F routines, and then calls assemble to put everything together.
- **polyamber96.F**: this subroutine contains templates for the amino acids for use with the Amber96 force field. This routine is called by buildprot.F
- **polyc19eef.F**: this subroutine contains templates for the amino acids for use with the C19eef force field. This routine is called by buildprot.F
- **polycharmm19.F**: this subroutine contains templates for the amino acids for use with the Charmm19 force field. This routine is called by buildprot.F
- **polycharmm22.F**: this subroutine contains templates for the amino acids for use with the Charmm22 force field. This routine is called by buildprot.F
- **polycharmm27.F**: this subroutine contains templates for the amino acids for use with the Charmm27 force field. This routine is called by buildprot.F.
- **polycharmm27na.F**: this subroutine contains templates for the nucleic acids for use with the Charmm27 force field. This routine is called by buildna.F.
- **polysafetycheck.F**: this subroutine checks to see if there is enough memory left in the molecule template arrays for the requested new atoms when building molecules using the poly*.F routines.
- **monomers.F**: contains shared information for each monomer that is used in the poly*.F routines.
- **assemble.F**: this subroutine takes atom names and a vibration map and determines all of the parameters needed for atoms, vibrations, bending angles, regular torsions, and angle-angle terms. It also determines the type of any improper torsion interactions.
- **createmolecule.F**: this subroutine is used with the readlammps functionality. This takes the connectivity map generated in readlammps and turns it into the molecule data structures needed for Towhee.
- **comparestruc.F**: this subroutine is used with the readlammps functionality. It compares two molecule data structures to see if they are identical.
- **getelement.F**: this subroutine is used with the readlammps functionality. This takes an elemental mass read in by readlammps and turns it into an element name for use with the pdb writer.
- **fffield.F**: this subroutine sets up all of the parameters needed for the external fields.
- **setmixterms.F**: this subroutine sets up all of nonbonded cross terms according to the mixing rule. It also converts all of the angles into radians.
- **checkstruc.F**: this subroutine sets up the bond-bond information, and it also performs extensive error checking on all of the molecules to make sure that they are set up in a consistent manner.
- **findrings.F**: this subroutine sets up data structures that are used when performing configurational-bias on

molecules which contain cyclic moieties, or on proteins.

- **initaverages.F**: this subroutine initializes all of the quantities that are tracked by the averages.F subroutine. Called from mainloop.F
- **initconf.F**: this subroutine sets up an initial conformation when linit is true. Otherwise the initial conformation is read from towhee_initial.
- **initialize.F**: this subroutine initializes many of the input variables. It is called from readinput before any variable are read from the input files.

Monte Carlo Moves and Move Support

- **volnpt.F**: this subroutine performs a volume change move on a single box and the move is accepted based on the energy change, number density, and the specified external pressure. The molecule positions are all scaled based upon their centers of mass so there are no changes to the bonded energies, although the intramolecular energy could change if you are using periodic boundary conditions with a large molecule. Note that you cannot use this move with a molecule that is bonded through the periodic boundary condition (like a zeolite).
- **volnvt.F**: this subroutine performs a volume change move on pair of boxes the move is accepted based on the energy change and the number densities. The molecule positions are all scaled based upon their centers of mass so there are no changes to the bonded energies, although the intramolecular energy could change if you are using periodic boundary conditions with a large molecule. Note that you cannot use this move with a molecule that is bonded through the periodic boundary condition (like a zeolite).
- **updatevolmaxdisp.F**: this subroutine periodically update the maximum volume displacements in an attempt to achieve the target acceptance rate set by the user.
- **swapmoves.F**: this subroutine performs all of the molecule swap moves. This includes 2-box configurational-bias molecule transfer, 1-box configurational-bias molecule reinsertion, 2-box rotational-bias molecule transfer, and the aggregation-volume-bias moves.
- **cbregrow.F**: this subroutine performs a configurational-bias molecule regrowth move. This is different from the intrabox molecule transfer move because at least one atom of the target molecule remains in the same location. This subroutine performs either the standard regrowth (pmcb), or a special backbone regrowth move (pmback) using the configurational-bias algorithm.
- **conrot.F**: this subroutine performs a concerted rotation move. This move is designed to change the torsional conformation of several interior atoms simultaneously while keeping the bond lengths and bending angles constant.
- **initloopclosure.F**: this subroutine initializes the variables used in a concerted rotation move.
- **loopclosure.F**: this subroutine performs the loop closure for the concerted rotation move.
- **pivot.F**: this subroutine performs the pivot move. This move is designed to rotate part of the molecule around a single torsional bond.
- **atomshift.F**: this subroutine performs the plane-shift and row-shift moves. These moves are designed to move groups of molecules at the same time.
- **tranatom.F**: this subroutine performs a translation move (in a random direction) upon one randomly selected atom of a molecule. The move is accepted based upon the energy change. This move is useful for large networked molecules (like zeolites), or for large molecules for which the configurational-bias moves have a low acceptance rate (like proteins).
- **trancom.F**: this subroutine attempts to move an entire molecule in a random direction. The move is accepted based upon the energy change.
- **rotate.F**: this subroutine attempts to rotate a molecule about a random axis that runs through the center of mass of the molecule. This move will not work properly for a molecule that is bonded through the periodic boundaries (like zeolites). The move is accepted based upon the energy change.
- **updatetrmaxdisp.F**: this subroutine periodically updates the maximum translation and rotation values in an attempt to achieve the target acceptance rate set in towhee_input.
- **reseteamrho.F**: this subroutine sets the EAM densities to the new values when a move is accepted using the Embedded Atom Method potential.
- **resetewald.F**: this subroutine sets the Ewald sum parameters either at the beginning of the simulation, or when a volume change is attempted.

Configurational-bias support routines

- **schedule.F**: this subroutine sets up the growth schedule for all of the moves that use the configurational-bias algorithm. This is the heart of the logic that controls the configurational-bias moves so you should be very careful about making changes in this subroutine.

- **rosenbluth.F**: this subroutine is the main control for all of the moves that use the configurational-bias algorithm. It grows either the new or old conformation of the molecule according to the plan determined in schedule.
- **getcbbond.F**: this subroutine constructs the bond lengths during a configurational-bias move.
- **getcbangle.F**: this subroutine constructs the bond bending angles during a configurational-bias move.
- **getcbdihed.F**: this subroutine constructs the dihedral angles during a configurational-bias move.
- **setcbdihed.F**: this subroutine constructs the logic needed to generate dihedral angles during a configurational-bias move.
- **dihedral_distribution.F**: this subroutine generates the dihedral trial conformations according to the appropriate arbitrary trial distribution.
- **getweight.F**: this subroutine turns a trial energy into a Rosenbluth weight. It makes sure that the weight returned does not go below machine precision by returning the weight as an integer scale factor and a double precision residual weight.
- **resetcbmc.F**: this subroutine resets all of the energy and weight variables at the start of a cbmc or rotational-bias move.
- **cone.F**: this subroutine is used to convert between Cartesian coordinates and spherical coordinates.
- **coneangle.F**: this subroutine takes two unit vectors in spherical coordinates and computes the angle between them.
- **sphere.F**: this subroutine randomly generates a unit vector on the unit sphere.
- **gaussian.F**: this function returns a number from a specified gaussian distribution.
- **gaussprob.F**: this function returns the probability density for taking a number from a specified gaussian distribution.
- **febias.F**: this function computes the bias that is used when performing a fixed-endpoint configurational-bias move.
- **findtarget.F**: this subroutine finds a target molecule for all of the swapmoves when they need to find a molecule that is in a certain region of space.
- **rotationmatrix.F**: this subroutine performs a rotation about the x, y, and z axes. This is used with both the rotation move, and with the rotational-bias move.
- **updatetwobondbias.F**: this subroutine periodically updates the variables associated with self-adapting two-bond fixed-endpoint biasing.
- **updatethreebondbias.F**: this subroutine periodically updates the variables associated with self-adapting three-bond fixed-endpoint biasing.

Energy and Pressure Routines

- **engtotal.F**: this subroutine computes the total energy for one of the simulation boxes. There is a special option for volume move calls to this routine that computes the total energy, except for the bonded intramolecular terms. All nonbonded inter- and intra-molecular terms are computed.
- **energy_change.F**: the subroutines in this file compute the energy change of a subset of atoms or molecules with the rest of the system.
- **energy.F**: this subroutines in this file compute the energy of a subset of atoms or molecules with the rest of the system.
- **engatom.F**: this subroutine computes the energy of a list of atoms with the rest of the atoms in one simulation box. This is used with the configurational-bias moves.
- **bondorder.F**: this subroutine computes the bond order for use with multibody force fields.
- **vbond.F**: this function computes a single bond vibration energy.
- **vbonbon.F**: this function computes a single Class 2 bond-bond energy.
- **vangle.F**: this function computes a single bending angle energy.
- **vangang.F**: this function computes a single Class 2 angle-angle energy.
- **vcoulomb.F**: this function computes real-space coulombic energy between two atoms.
- **vfield.F**: this function computes the interaction energy of an atom with all of the external fields.
- **vimproper.F**: this function computes an improper torsion energy.
- **vtorsion.F**: this function computes a regular torsion energy.
- **vtwobody.F**: this function computes the van der Waals energy between two atoms.
- **vthreebody.F**: this function computes a three-body term between three atoms which are not bonded to each other but interact via a three body force field

- **vembed.F**: this function computes the embedding energy for the EAM potential
- **veefone.F**: this function computes the solvation energy using the EEF1 implicit solvent.
- **vsasa.F**: this function computes the solvation energy using the SASA implicit solvent.
- **fielddft.F**: this subroutine controls the computation of the solvation free energies using the molecular density functional theory program Tramonto.
- **stresstensor.F**: this subroutine computes the stress tensor and pressure in a simulation box.
- **wtwobody.F**: this subroutine computes the intermolecular pair virial function divided by r^2 as defined by equation 2.60 in Allen and Tildesley
- **recip.F**: this subroutine updates the stored sums that are used to reduce the cost of the Ewald sum for single molecule moves.
- **recippress.F**: this subroutine computes the reciprocal space contribution to the pressure using the Ewald sum
- **recipsum.F**: this subroutine computes the reciprocal space portion of the Ewald sum for an entire simulation box.
- **tail.F**: this subroutine calculates the values needed to compute analytical tail corrections to the energy and pressure.
- **interpolate.F**: this file contains several functions that interpolate the cubic splines used with the embedded atom potential.
- **spline.F**: this subroutine sets up the cubic splines that are used in the lookup tables for the embedded-atom potential.

Output and Averages

- **averages.F**: this subroutine keeps track of the data structures used to perform the regular and block averaging during the simulation.
- **accumulateaverages.F**: this subroutine controls the averaging operations that occur at the end each move.
- **pupdate.F**: this subroutine controls the averaging operations that occur at the end of a block.
- **openfile.F**: this subroutine opens up a file in the appropriate directory.
- **rwconf.F**: this subroutine either inputs a configuration file from towhee_initial or outputs a configuration file to either towhee_backup or towhee_final.
- **writeaverages.F**: this subroutine outputs all of the total average and block average information at the end of the run.
- **writedynamo.F**: this subroutine outputs the embedded atom method potential using the dynamo format.
- **writeintro.F**: this subroutine outputs information about the Towhee project and the GNU public license at the start of the run. Called by mainloop.F.
- **writelammps.F**: this subroutine outputs files of the final configuration that are suitable for use as inputs to the molecular dynamics code LAMMPS.
- **writemovie.F**: this subroutine outputs snapshots of the simulation to the towhee_movie file for later analysis with the analyse_movie utility.
- **writepdb.F**: this subroutine outputs files of the final configuration in the pdb format. These are useful for visualization of the simulation using a tool such as RasMol.
- **writeruntime.F**: this subroutine writes run-time information to standard output with a frequency that is set by the **printfreq** variable.
- **writetowhee.F**: this subroutine outputs the towhee_altinp file. This file is suitable for use as a towhee_input file in subsequent simulations.
- **writetramonto.F**: this subroutine outputs files of the final configuration that are suitable for use as inputs to the classical density functional code Tramonto.
- **writeangang.F**: this subroutine outputs all of the constants for all of the angle-angle terms used in the simulation.
- **writeangle.F**: this subroutine outputs all of the constants for all of the angle terms used in the simulation.
- **writebond.F**: this subroutine outputs all of the constants for all of the bond terms used in the simulation.
- **writeimproper.F**: this subroutine outputs all of the constants for all of the improper torsion terms used in the simulation.
- **writetorsion.F**: this subroutine outputs all of the constants for all of the regular torsion terms used in the simulation.
- **writenonbond.F**: this subroutine outputs the constants for all of the nonbonded terms used in the simulation.

Global Data, Coordinates and Periodic Boundaries

- **arbtocart.F**: transforms from the arbitrary coordinate system (based on the hmatrix) into the Cartesian coordinate system.
- **carttoarb.F**: transforms from the Cartesian coordinate system into the arbitrary coordinate system.
- **checkhmatrix.F**: checks to make sure that all three vectors of the hmatrix have an angle of at least 45 degrees between them. This is needed as a safety check on the simplified version of the minimum image convention.
- **inverthmatrix.F**: inverts the hmatrix and computes the boxvolume. The inverse hmatrix is used to transform between the Cartesian and arbitrary coordinate systems.
- **getnbtype.F**: this subroutine determines the array index for the interactions between two types of atoms.
- **globaldata.F**: this file contains many subroutines that are replacing the previous global data structures (contained in globalf.h) with localized data that is accessed via subroutine calls.
- **globalaccess.F**: this file contains functions and subroutines that make it easier to access the data structures that are now contained in globaldata.F.
- **mimage.F**: this subroutine enforces the minimum image convention on a set of distances.
- **minboxlength.F**: this subroutine returns the minimum box length.
- **maxboxlength.F**: this subroutine returns the maximum box length.
- **putarbinbox.F**: this subroutine takes a set of arbitrary coordinates and puts them back into the central box image.
- **putcartinbox.F**: this subroutine takes a set of Cartesian coordinates and puts them back into the central box image.
- **uniformbox.F**: this subroutine generates the Cartesian coordinates of an atom that is selected uniformly from the entire central simulation box.

Force Field Fitting and Database Routines

- **readdatabase.F**: this subroutine reads information from the database_input date file and evaluates the database of structures against a trial force field. It is also capable of some simple optimization.
- **leastsquares.F**: this subroutine performs a least squares fit on the arrays passed into it.
- **sdb_config.F**: one of the support subroutines for parsing the sdb database format.
- **sdb_rdstruct.F**: one of the support subroutines for parsing the sdb database format.
- **sdb_rdtoend.F**: one of the support subroutines for parsing the sdb database format.
- **sdb_strget.F**: one of the support subroutines for parsing the sdb database format.
- **sdb_strpars.F**: one of the support subroutines for parsing the sdb database format.

Other Routines

- **arccos.F**: this is a safe implementation of the standard FORTRAN function dacos. It does not crash when taking the anti-cosine of a number that is just a bit larger than 1.0 or a bit smaller than -1.0.
- **crossproduct.F**: this subroutine computes the cross product of two vectors.
- **ctrmas.F**: this subroutine computes the center-of-mass of molecules and also the maximum distance from any atom to the center-of-mass.
- **distance.F**: this function computes the distance given a three dimensional vector.
- **dotproduct.F**: this function computes the normalized dot product of two vectors.
- **expon.F**: this function computes exponential of a number in a safe way so that if the value inside the exponential would result in a number that is less than SMALLEST, the function instead returns zero. Thus, this function will never result in an underflow error, although an overflow is still possible as there are other checks in the code to prevent an overflow.
- **internal.F**: this file contains subroutines that are normally handled by intrinsic functions, but occasionally need explicit support on machines that do not have the appropriate intrinsic libraries.
- **linclude.F**: this file contains a subroutine (setinclude) and a function (linclude) which are used to determine if two atoms on the same molecule have nonbonded interactions.
- **minimizer.F**: this subroutine performs a minimization of the coordinates and box dimensions.
- **problem.F**: this subroutine is called when there is a problem in the code and we need to exit. Currently it only sets a logical, but in the future it will handle the graceful exit of the code during a multiprocessor simulation.
- **random.F**: a file which contains one function (random) which controls the generation of random numbers. There are other functions and subroutines in this file which generate the random numbers.

forcefield Program Routines (not needed to compile towhee)

- **createff.F**: the main driver for the forcefield program.
- **ffcheck.F**: this subroutine performs bounds checking on the force field generation routines.
- **ffackl2004.F**: this subroutine sets all of the parameters needed to build the Ackland *et al.* 2004 force field.
- **ffalavi2005.F**: this subroutine sets all of the parameters needed to build the Alavi *et al.* 2005 force field.
- **ffamber96.F**: this subroutine sets all of the parameters needed to build the Amber96 force field.
- **ffaqvist.F**: this subroutine sets all of the parameters needed to build the Aqvist force field.
- **ffc19eef1.F**: this subroutine sets all of the parameters needed to build the C19eef1 force field.
- **ffc19sasa.F**: this subroutine sets all of the parameters needed to build the C19sasa force field.
- **ffcatlowfaux.F**: this subroutine sets all of the parameters needed to build the CatlowFaux force field.
- **ffcharmm19.F**: this subroutine sets all of the parameters needed to build the Charmm19 force field.
- **ffc27rigid.F**: this subroutine sets all of the parameters needed to build the C27rigid force field.
- **ffcharmm22.F**: this subroutine sets all of the parameters needed to build the Charmm22 force field.
- **ffcharmm22fe.F**: this subroutine sets all of the parameters needed to build the Charmm22fe supplemental force field.
- **ffcharmm27.F**: this subroutine sets all of the parameters needed to build the Charmm27 force field.
- **ffcharmm27x.F**: this subroutine sets all of the parameters needed to build the Charmm27x force field supplement.
- **ffclayff.F**: this subroutine sets all of the parameters needed to build the ClayFF force field.
- **ffcompass.F**: this subroutine sets all of the parameters needed to build the COMPASSv1 force field.
- **ffcoon1987.F**: this subroutine sets all of the parameters needed to build the Coon1987 force field.
- **ffcui1998.F**: this subroutine sets all of the parameters needed to build the Cui *et al.* 1998 force field.
- **ffcui2002.F**: this subroutine sets all of the parameters needed to build the Cui and Elliott 2002 force field.
- **ffdacnisua.F**: this subroutine sets all of the parameters needed to build the DACNIS-UA force field.
- **ffdick1994.F**: this subroutine sets all of the parameters needed to build the Dick and Ritchie 1994 force field.
- **ffding1986.F**: this subroutine sets all of the parameters needed to build the Ding1986 force field.
- **ffdreiding.F**: this subroutine sets all of the parameters needed to build the DREIDING force field.
- **ffdubb2004.F**: this subroutine sets all of the parameters needed to build the Dubb2004 force field.
- **ffelli2002.F**: this subroutine sets all of the parameters needed to build the Elliott 2002 force field.
- **ffepm.F**: this subroutine sets all of the parameters needed to build the EPM force field.
- **fffris2003.F**: this subroutine sets all of the parameters needed to build the Fris2003 force field.
- **ffgala1994.F**: this subroutine sets all of the parameters needed to build the Gala1994 force field.
- **ffgordon.F**: this subroutine sets all of the parameters needed to build the Gordon force field.
- **ffgromos43a1.F**: this subroutine sets all of the parameters needed to build the Gromos43A1 force field.
- **ffhardsphere.F**: this subroutine sets all of the parameters needed to build the HardSphere force field.
- **ffkfvbvs.F**: this subroutine sets all of the parameters needed to build the KFvBvS force field.
- **fflast1993.F**: this subroutine sets all of the parameters needed to build the Last1993 force field.
- **fflgm.F**: this subroutine sets all of the parameters needed to build the LGM force field.
- **ffljium.F**: this subroutine sets all of the parameters needed to build the LJium force field.
- **ffmend2003.F**: this subroutine sets all of the parameters needed to build the Mendelev *et al.* 2003 force field.
- **ffmgmstereo.F**: this subroutine sets all of the parameters needed to build the MGM stereochemistry enforcer force field supplement.
- **ffmm2.F**: this subroutine sets all of the parameters needed to build the MM2 force field.
- **ffmorrow2002.F**: this subroutine sets all of the parameters needed to build the Morrow and Maginn 2002 force field.
- **ffnerdv1.F**: this subroutine sets all of the parameters needed to build the NERD Version 1 force field.
- **ffnerdv2.F**: this subroutine sets all of the parameters needed to build the NERD Version 2 force field.
- **ffnerdv3.F**: this subroutine sets all of the parameters needed to build the NERD Version 3 force field.
- **ffopls2001.F**: this subroutine sets all of the parameters needed to build the OPLS-2001 force field.
- **ffoplsaa.F**: this subroutine sets all of the parameters needed to build the OPLS-aa force field.
- **ffoplsua.F**: this subroutine sets all of the parameters needed to build the OPLS-ua force field.
- **ffpana1989.F**: this subroutine sets all of the parameters needed to build the Panagiotopoulos 1989 force field.
- **ffpmf.F**: this subroutine reads in radial distribution function and converts it into a potential of mean force

tabulated potential.
- **ffpotter1997.F**: this subroutine sets all of the parameters needed to build the Potter *et al.* 1997 force field.
- **ffqmff_viii.F**: this subroutine contains a partial implementation of the QMFF-VIII force field.
- **ffreadcharmmfile.F**: this subroutine reads in a native Charmm force field file and populates the Towhee force field data structure.
- **ffreadsetflfile.F**: this subroutine reads in an EAM setfl file and populates the Towhee force field data structure.
- **ffshah2004.F**: this subroutine sets all of the parameters needed to build the Shah and Maginn 2004 force field.
- **ffshukla1987.F**: this subroutine sets all of the parameters needed to build the Shukla 1987 force field.
- **ffsks.F**: this subroutine sets all of the parameters needed to build the SKS force field.
- **ffsmith1994.F**: this subroutine sets all of the parameters needed to build the Smith and Dang 1994 force field.
- **ffsmmkmain.F**: this subroutine sets all of the parameters needed to build the SMMK main force field.
- **ffsmmknaip.F**: this subroutine sets all of the parameters needed to build the SMMK Note Added in Proof force field.
- **ffspce.F**: this subroutine sets all of the parameters needed to build the SPC-E force field.
- **ffsquarewell.F**: this subroutine sets all of the parameters needed to build the SquareWell force field.
- **ffstil1985.F**: this subroutine sets all of the parameters needed to build the Stil1985 force field.
- **ffsum2003.F**: this subroutine sets all of the parameters needed to build the Sum2003 force field.
- **fftele1987.F**: this subroutine sets all of the parameters needed to build the Tele1987 force field.
- **fftip3p.F**: this subroutine sets all of the parameters needed to build the TIP3P force field.
- **fftip4p.F**: this subroutine sets all of the parameters needed to build the TIP4P force field.
- **fftip5p.F**: this subroutine sets all of the parameters needed to build the TIP5P force field.
- **fftrappeeh.F**: this subroutine sets all of the parameters needed to build the TraPPE-EH force field.
- **fftrappeua.F**: this subroutine sets all of the parameters needed to build the TraPPE-UA force field.
- **fftrappeuaf.F**: this subroutine sets all of the parameters needed to build the TraPPE-UA flexible force field.
- **ffuff.F**: this subroutine sets all of the parameters needed to build the UFF force field.
- **ffunlu2004.F**: this subroutine contains a partial implementation of the Unlu and Elliott 2004 force field.
- **ffvega1992.F**: this subroutine sets all of the parameters needed to build the Vega *et al.* 1992 force field.
- **ffvink2001.F**: this subroutine sets all of the parameters needed to build the Vink2001 force field.
- **ffwalt2001.F**: this subroutine sets all of the parameters needed to build the Walt2001 force field.
- **ffweiner1984.F**: this subroutine sets all of the parameters needed to build the Weiner *et al.* 1984 force field.
- **ffweiner1986.F**: this subroutine sets all of the parameters needed to build the Weiner *et al.* 1986 force field.

The .c files

C routines

- **towhee.c**: This is where everything begins (main). If using the MPI version of towhee this handles the initialization of MPI and also reads from the towhee_parallel file. Otherwise it just handles the command line variables sent to Towhee and then calls **mainloop.F**.
- **control.c**: This set of routines is used to handle some of the MPI calls and to serve as an interface between Towhee and the Tramonto codes.
- **jobfarm.c**: This set of routines handles the MPI calls required to run a parallel simulation that farms out a list of jobs over several processors, where each processor runs a single job at a time.
- **rex.c**: Implementation of Replica Exchange functionality. Replica Exchange requires MPI to run. While functional, it is currently at an early stage of development.
- **sturm.c**: These routines handle much of the mathematics required for the concerted rotation moves.

Bug Reports

In a code of this magnitude there will invariably be bugs from time to time. Hopefully all of the big ones are out of the code, but there is a continual process of adding features (rebugging) and testing the features (debugging). If you do find a problems with the code, please let us know so we can fix it for the benefit of everyone who is using the program. The more detail you can provide in your bug report the better. If you have a specific case that crashes then sending copies of the input files will greatly expedite the debugging process. We are now using the Bug Tracking software at SourceForge. To submit a bug just head to the Towhee Bug Tracker site and click on the "submit new" button. The best option is for you to log into SourceForge (creating an account is free) and use that account to submit the bug, in which case it will automatically notify you of the progress resolving the bug. Alternatively, you can send an email about your bug to towhee-bugs@lists.sourceforge.net. Your posting to this list is automatically held until we can verify it is not spam.

Return to the main towhee web page

Send comments to: Marcus G. Martin

Last updated: January 03, 2007

MCCCS Towhee (example manual)

Overview

This section describes the examples included with the current download version of Towhee and gives some tips for getting started with these examples.

General Tips

All of the example files are found in the Examples directory that comes with the default download package. You will find a file named answer_current in each directory and that contains the testing output for each example. Tests for version 3.14.9 and later were run using gcc version 3.2.2 with the FFLAGS=-fbounds-check option in the configure script. While this is a great option for debugging we suggest letting the configure option determine the optimization level automatically when compiling the code for production runs. The -O2 level of optimization results in noticable improvements in the code speed. Starting with version 4.13.1, the compile-time flag --enable-safe-compare attempts to minimize differences in the output between platforms, but should not be used for production runs.

Each of the towhee_input files has to specify the complete path of the force field files it wishes to utilize. The default setting is something like `/towheebase/ForceFields/towhee_ff_Charmm22` where Charmm22 would be the force field you are using. You can either change the `/towheebase` section to be the directory path for the ForceFields file, or you can set a symbolic link for `/towheebase` using the standard unix command

```
ln -s /Your/Actual/Directory /towheebase
```

As an alternative to either manually modifying the ForceFields files or creating symbolic links, execute the `relocate_examples.pl` script in the Examples directory. This script copies all the examples to a directory named `test` in the towhee root directory, and will simultaneously rewrite all references to `/towheebase` appropriately. You can then execute examples in the normal fashion from the `test` directory.

To run the code you need to be in one of the example directories and then execute a command similar to `/towheebase/Source/towhee` where again `/towheebase` is replaced by your actual directory structure.

Descriptive List of Examples

- Amber_IsoPropanol
  A two box NVT Gibbs ensemble that demonstrates the setup for determining single-component vapor-liquid coexistence. Uses the Lennard-Jones potential and the Amber96 force field.
- Amber_Villin
  A single box simulation of a short peptide chain in vacuum to demonstrate the use of the protien builder functionality. Uses the Lennard-Jones potential and the Amber96 force field.
- Au_Cu_Switch
  A single box simulation demonstrating the use of the center-of-mass switch move for a mixture of monatomic metal molecules with an EAM potential.
- AVB1_Methane
  A single box simulation that demonstrates the use of the agregation-volume-bias move type 1. Uses the Lennard-Jones potential and the TraPPE-UA force field.
- Catlow_Zeolite_4a
  A two box, multi-component simulation demonstrating the setup for computing adsorption isotherms in porous materials. Uses the combined exponential-6 and Lennard-Jones potential and the Catlow force field.
- Charmm19_ubiquitin
  A single box simulation that demonstrates the use of the Charmm19-EEF1 potential with implicit water solvation for a protein.
- Charmm22_Ethanethiol
  A single box simulation that demonstrates how to compute a liquid density using the isobaric-isothermal ensemble. Uses the Lennard-Jones potential and the Charmm22 force field.
- Charmm27_Heme

A single box simulation that shows how to set up a Heme group in Charmm27 using the atom builder.

- Charmm27_NA
  A single box simulation that shows how to set up a nucleic acid in Charmm27 using the nucleic acid builder.
- Charmm27_Polyalanine
  A single box single molecule simulation that demonstrates the use of the 'helix cbmc' initialization option. This builds a polypeptide with the C_a atoms in a helix of specified radius and angle and then grows the rest of the atom using configurational-bias.
- Compass_Methanol
  A single box, single molecule simulation that demonstrates how to compute the energy for a single molecule in vacuum by disabling the Ewald sum. Uses the 9-6 potentials and the Compass force field.
- Convert/LAMMPS_class2
  Demonstrates the conversion of LAMMPS files into Towhee input files. To perform this conversion change the first line of the towhee_input file to the 'LAMMPS' inputformat instead of the 'Towhee' inputformat. Running Towhee will create files suitable for a real Towhee run. Copy these as instructed by the code and then run again to perform a Towhee simulation with the new files. Uses the 9-6 force field.
- Convert/LAMMPS_decane
  Demonstrates the conversion of LAMMPS files into Towhee input files. To perform this conversion change the first line of the towhee_input file to the 'LAMMPS' inputformat instead of the 'Towhee' inputformat. Running Towhee will create files suitable for a real Towhee run. Copy these as instructed by the code and then run again to perform a Towhee simulation with the new files. Uses the Lennard-Jones force field.
- Convert/LAMMPS_lc
  Demonstrates the conversion of LAMMPS files into Towhee input files. To perform this conversion change the first line of the towhee_input file to the 'LAMMPS' inputformat instead of the 'Towhee' inputformat. Running Towhee will create files suitable for a real Towhee run. Copy these as instructed by the code and then run again to perform a Towhee simulation with the new files. Uses the Lennard-Jones force field.
- Cu_Pb_EAM
  A single box simulation using the embedded atom potential for a mixture of copper and lead.
- Cu_VLE
  A two box simulation using the embedded atom potential to compute vapor-liquid coexistence for copper.
- DFT_Field
  Test case that combines Towhee with the Tramonto package as an implicit solvent. The Tramonto package is not yet publicly available.
- Dick1994_PETN
  An example that shows how to construct an initial system by replicating a unit cell. Creates a PETN solid using the Dick and Ritchie 1994 force field.
- DREIDING
  An example that shows how to minimize an initial structure by using single atom translations and a low temperature. Uses the DREIDING force field.
- Energy_Biasing
  An example that shows how to set up the input files to create a towhee_map file to utilize energy biasing for adsorption of molecules in porous materials. Uses the Dubb2004 force field.
- EPM_VLCC
  An example of a single-component vapor-liquid coexistence simulation taken from Table 4 of Harris and Yung 1995. Uses the Lennard-Jones potential and the EPM2 force field.
- FENE_Hexamer
  An example using the FENE bond potential for a simple Lennard-Jones hexamer.
- Formamide_Scaled
  A one box, two component simulation that demonstrates how to perform a simulation with a 'Scaled Lennard-Jones' classical potential.
- Fris_Walls
  A one box, single component simulation that demonstrates how to perform a simulation between Lennard-Jones walls with additional hard walls and an "umbrella" potential.
- Gordon
  A single box, single component simulation just to check that the Gordon n-6 potential gives the same

energies and pressures as the Lennard-Jones potential when n is set to 12.

- Gromos_Isobutane
  A two box, single component simulation that demonstrates how to perform a single-component vapor-liquid coexistence curve. Uses the Lennard-Jones potential and the Gromos43A1 force field.
- Gromos_Methylpropylsulfide
  A single box, single component simulation that demonstrates how to determine liquid densities. Uses the Lennard-Jones potential and the Gromos43A1 force field.
- Hard_Sphere
  A single box, multicomponent simulation of hard spheres. Uses the Hard Sphere potential and force field.
- Henry Law
  A single box, multicomponent simulation designed to measure the Henry's Law coefficient of four small gases in a liquid ethanol solvent. This example reproduces the results from the winner of the Second Industrial Fluids Simulation Challenge Problem 2 involving the Henry's law coefficient for methane, O_2, N_2 and CO2 in ethanol using a combination of the TraPPE-UA, TraPPE-EH, and Coon1987 force fields. I suggest using an **nstep** value of 50000 and a **blocksize** of 10000 for production runs of this system as the chemical potential is challenging to compute precisely with Widom insertion.
- Histogram
  This directory contains three subdirectories (Phase, PVT, Weights) that each illustrate one aspect of the analyse_histogram routine. This routine is used to process the output from grand canonical simulations using the Histogram Reweighting method. Please see the README.histogram file in the Examples/Histogram directory for more information.
- Ideal_Chain
  A single box, single component demonstration of the configurational-bias algorithm for a flexible ideal chain in the isothermal-isobaric ensemble. Uses the Hard Sphere potential with the zero diameter hard sphere force field.
- MM2_Ethane
  A two box simulation of ethane using the MM2 force field.
- NaCl_1x1x1
  A single box computation of the Madelung constant to verify the accuracy of the Ewald summation. The correct answer is 1.747558 (dimensionless) and this example allows the user to modify the parameters of the Ewald sum and see how this affects the accuracy of the total coulombic energy.
- OPLS_Propanamide
  A single box single component liquid density determination. Uses the Lennard-Jones potential and the OPLS-aa force field.
- Parallel_Test
  An example of how to use the towhee_parallel file with the mpitowhee version of the code to run multiple simulations on one or more processors.
- Potter_CF2H2
  An example of the 'LB plus manual' classical mixing rule applied to reproduce a two box vapor-liquid coexistence point using the Potter *et al.* 1997 force field.
- Shukla_Gasses
  An example of the 'Skukla' classical mixing rule for a trivial simulation containing four different small molecules in the gas phase using the Shukla 1987 force field.
- SKS_Pentane
  An example from the literature to illustrate the calculation of vapor-liquid coexistence using the SKS potential for *n*-alkanes. Also serves as a test case for rotational-bias and configurational-bias 2 box swap moves.
- Small_Peptide
  An example of how to create a small peptide with the initial structure generated via CBMC.
- SMMKmain_2244688nonane
  An example of how to create an initial structure of 2,2,4,4,6,8,8-heptamethyl nonane using the SMMKmain forcefield. Also serves as a test of the special one-five interactions.
- SMMKnaip_Ethylpentane
  An example of how to create an initial system suitable for 2-box Gibbs ensemble determination of vapor-

liquid coexistence of 3-ethylpentane using the SMMKnaip forcefield. Equilibrating this starting structure (likely several runs of 10,000 cycles) should result in agreement with the densities reported in Table 3 of Siepmann *et al.* 1997.

- Solid_LJium
  A simulation of solid Lennard-Jonesium to demonstrate the use of the plane shift and row shift moves.
- Square_Well_Chain
  A single box, single component demonstration of the configurational-bias algorithm for a tangent sphere square well chain in the canonical ensemble. Uses the Square Well potential with the generic SquareWell force field.
- Steele_Wall
  A single box, single component demonstration of the Steele surface potential with the Last1993 Lennard-Jones potential in the grand canonical ensemble.
- SW_Silicon
  An example of how to scan a database of solid structures and compute the energies. This example can also be modified to scan multiple parameters of a force field and compute the RMS differences from the database entries. Uses the Stillinger-Weber potential and the Stillinger-Weber Silicon force field.
- TMMC
  Examples (Lennard-Jones and SPC/E Water) for performing grand-canonical transition-matrix Monte Carlo (TMMC).
- TraPPE_Isomers
  Two butene isomers that demonstrate the builder features that allow the user to specify the difference between cis and trans dihedrals using the TraPPE-UA forcefield.
- TraPPE_Molecules
  A batch of single molecule structures to test the assembler and bond increment method for the TraPPE-UA force field and to test the center-of-mass switch move for polyatomic molecules.
- TraPPE_Pentane
  An example of a single-component vapor-liquid coexistence simulation. Uses the Lennard-Jones potential and the TraPPE-UA force field.
- Triglycerid
  A setup example showing how to build the simple triglyceride model from the Sum *et al.* paper using the Sum2003 forcefield.
- UFF
  An example that shows how to minimize an initial structure by using single atom translations and a low temperature. Uses the UFF force field.
- uVT_Ethane
  An example of a grand canonical ensemble simulation with the output flags set to generate towhee_histogram files.
- VLCC_Fit
  A set of scripts and files the provides and example of how to use a sequence of towhee_vlcc output files, and some experimental data to compute the predicted critical temperature and critical density for a single component system. **run_fitvlcc** is the script that copies files and calls the `Utils/fitvlcc.x` program. **plot** is a script that uses the freely available xmgrace program to plot the results.
- Wall_Water
  An example of use of fields in Towhee. Demonstrates water between two hard walls.
- Walt2001_Nanotube
  An example showcasing the nanotube builder functionallity. Grand canonical ensemble simulation with a single nanotube and the chemical potential set to fill the simulation box with water.
- Weiner1984
  A two box simulation using the Weiner *et al.* 1984 force field.

Return to the main towhee web page

Send comments to: Marcus G. Martin

Last updated: August 08, 2008

MCCCS Towhee (Utility Summary)

Overview

This section summarizes the utility programs that are included with the Towhee code distribution (generally in the **Utils** directory). It also contains links to detailed information explaining how to use each of these programs.

Post-Processing Data Analysis Utilities

analyse_histogram
: This program is designed for use in conjunction with the **towhee_histogram** files and is used to analyze the histogram data generated by the grand canonical ensemble.

analyse_movie
: This program is designed for use in conjunction with the Towhee MCCCS file and is used to compute quantities from the **towhee_movie** file.

fitcoex
: This program replaces the old fitvlcc utility and is designed for use in conjunction with the **towhee_vlcc** output files in order to compute the critical temperature and critical density from a sequence of single-component vapour-liquid simulations.

parse_vlcc_plots
: This script is designed for use in conjunction with the standard towhee output file used in a single-component vapour-liquid simulation and it extracts the snapshot and block average information from that file in order to create a series of plottable files.

Pre-Processing File Conversion Utilities

charmm2pdb
: This program is designed to translate the polypeptide pdb files generated by Charmm into the standard pdb format to allow further processing via the pdb2towhee routine.

faux2towhee
: This program is designed to translate some zeolite coordinate files that were kindly provided to me by David Faux into a format suitable for Towhee. This utility is unlikely to be of use to anyone else, but may serve as an example for other one-time file translations.

forcefield
: This program uses many of the main Towhee subroutines along with several additional subroutines to convert a forcefield from a different format into the Towhee Force Field Format. It is capable of generating almost all of the force field files included in the ForceFields directory of the Towhee distribution.

pdb2towhee
: This program is designed to translate the standard pdb files into appropriate Towhee files for starting a simulation. In all cases it will output a **towhee_coords** file full of the system coordinates. In certain cases it will output a **towhee_altinp** file with partial information required to set up a polypeptide molecule template.

rdf2pmfpair
: This program converts a radial distribution function into a potential of mean force. The potential of mean force file is then suitable for conversion into a tabulated pair potential using the forcefield program.

unitcell
: This program is used to take a towhee_coords file and duplicate it any number of times in the x, y, and z dimensions in order to perform a larger simulation.

xmd2towhee
: This program converts EAM force field files that are in the XMD format into a format suitable for use in Towhee.

xyz2towhee
: This program converts simple xyz format files into a series of files that can be used to construct a **towhee_coords** file for Towhee.

Return to the main towhee web page

Send comments to: Marcus G. Martin
Last updated: December 14, 2006

MCCCS Towhee (Developer Manual)

Overview

This section covers the basic information that is required in order to become a Towhee developer and also some suggested practices and overall guidelines for working with the Towhee development team.

Becoming a Developer

Towhee is an open-source project so anyone is welcome to download the package, including all of the source code, and start modifying it for their own purposes. Here we are concerned with people who want to develop Towhee in conjunction with the current team of developers. One way to help is to report any problems you discover in the code to the developers via the towhee-bugs@lists.sourceforge.net mailing list. This is also a great way to communicate minor code changes, especially those that fix bugs, to the developers for them to include in the next release.

Those wishing to make substantial changes to the code, such as implementing new algorithms, are better served by joining the Towhee developer team in order to gain full access to the CVS repositories at SourceForge. If you are interested then you need to sign up for a free username at SourceForge (http://sourceforge.net/), and send an email indicating your interest to Marcus G. Martin.

CVS Access

In order to get started using CVS you will need to checkout the current version of the code into a workspace on one of your machines. Directions for checking out the code are found by following the CVS link from the Towhee SourceForge site. Make sure to follow the developer access directions, and not the anonymous access directions. You first need to set the **CVS_RSH** environment variable to **ssh**. Then you can checkout the code using

```
cvs -z3 -d:ext:developername@towhee.cvs.sourceforge.net:/cvsroot/towhee co -P towhee
```

where *developername* is replaced by your SourceForge user name. This command creates a directory named **towhee** that contains all of the Towhee files and directories.

For a complete discussion of CVS see a resource like the Open Source Development with CVS. A few commands are especially useful and are mentioned here.

```
cvs update
```

checks your workbox copy of the code against the repository and attempts to reconcile the two copies by bringing your workbox version up to date. Pay special attention to any "conflict" messages during the update as these indicate a problem bringing the workbox up to date and a note indicating the conflict is placed into all files that contained a conflict.

```
cvs commit
```

is used to check your changes back into the repository. During this process you are given an opportunity to make some comments about your changes and it is helpful if you can summarize your changes in a couple of sentences.

Test Suite

The Towhee example manual also doubles as the test suite for the code. Before you commit changes to the repository you should build the **towhee** executable, using the **--enable-safe-compare** configure flag and run it through the test suite in order to make sure you have not caused any unintentional damage to the rest of the code while implementing your new features. This is done using the **./run_serial_test** command in the Examples directory. This sequentially works through the examples producing files named **answer_new** in each of the directories. At the conclusion of the script it creates a file named **test_report** that contains the Unix diff of all of the **answer_new** files with their respective **answer_current** files. Once you are satisfied that the differences between the files are an obvious and intended result of your changes then you can copy all of the **answer_new** files to **answer_current** files using the **./reset_new_to_current** script in the Examples directory.

If you are working on parallel programming features of Towhee then you will also want to run the

./run_para_test script in the Examples directory. This script then called the **./execute_parallel** script that is in the Parallel_Test directory. The **./execute_parallel** script is no longer included in the distribution because it is a machine specific script that executes a parallel job in that directory. If you are on a machine that allows direct submission of parallel jobs then your **./execute_parallel** script would look something like

```
mpirun -np 2 /towheebase/Source/towhee
```

This will execute a job that utilizes the **towhee_parallel** file in the Parallel_Test directory to run a jobfarm that executes the test suite jobs to produce output files named **par_answer_current**. Running the **./run_para_diff** script in the Examples directory produces the **para_test_report** file that contains the diffs from all of the **par_answer_current** and their respective **answer_current** files. These files should be identical so the **para_test_report** file should only contain the directory names of the test suite.

Return to the main towhee web page

Send comments to: Marcus G. Martin

Last updated: December 14, 2006

MCCCS Towhee (download)

Overview

This section describes the steps required to download a version of the MCCCS Towhee program. Contact towhee-bugs@lists.sourceforge.net if you have any problems downloading this software.

Subscribing to the towhee-users mailing list

Information about significant feature enhancements and bug fixes is broadcast on the towhee-users mailing list. In addition, this serves as a forum for users to ask questions about the use of the Towhee code. If you wish to subscribe then you can do so on the SourceForge towhee-users website at http://lists.sourceforge.net/lists/listinfo/towhee-users.

Download instructions for a Traditional Download

The current version of the Towhee code is available for download from the MCCCS Towhee SourceForge website. The link to that website is http://sourceforge.net/projects/towhee/.
Once you have downloaded the code you need to GNU unzip it using the following commands on a unix machine

- gunzip towhee_current.tar.gz
- tar -xf towhee_current.tar

This will create a directory named

towhee-xx.yy.zz

where xx.yy.zz will be the version number of the download.

Download instructions using CVS on SourceForge

This option is only suggested for those who are programming geeks (like myself) and are interested in joining in on the development of Towhee.. The CVS repository for the code is located on SourceForge and can be accessed by following the instructions at Project:MCCCS Towhee:CVS on SourceForge.
Those interested in becoming a developer of the code and gaining CVS access on SourceForge should contact Marcus G. Martin. It is not necessary to be an offical developer in order to contribute to the project, just an added bonus for those who plan to work extensively on one or more feature enhancements.

Compiling and Running Towhee

Please see the Towhee code manual for information about compiling or modifying the code.
See the Towhee users manual to learn how to set up the input files and run the code.

Return to the main towhee web page

Send comments to: Marcus G. Martin
Last updated: September 22, 2005

MCCCS Towhee (Monte Carlo)

Overview

This section provides a quick overview of statistical mechanics and discusses the reasons why somebody might want to perform a Monte Carlo molecular simulation. This was adapted from the Ph.D. thesis of Marcus Martin.

Why would you perform a Molecular Simulation?

Statistical mechanics is a theory that takes as its input the total energy of a system of molecules as a function of their positions and momenta and yields as an output any thermodynamic property of interest. This involves an integral in which momenta (kinetic energy) and positions (potential energy) are easily separated and the part involving the momenta can be integrated analytically. Unfortunately, except for some very trivial (and not realistic) functional forms for the potential energy, it is not possible to analytically integrate the part involving the positions, and a numeric solution is useless because a derivative of the original integral must be taken in order to compute most of the thermodynamic properties.

Rewriting this problem in the language of probability theory allows some (but not all) of the thermodynamic properties to be computed as the expected value or variance of some distribution. For the canonical ensemble, where we have a constant number of molecules (**N**), a constant total volume (**V**), and a constant temperature (**T**), the expected value of any observable *j* can be written

$$E[j] = \frac{\int_{\mathbf{r}} j \times \exp[-\beta U(\mathbf{r})] d\mathbf{r}}{\int_{\mathbf{r}} \exp[-\beta U(\mathbf{r})] d\mathbf{r}} \qquad Eq.1$$

where **r** is the set of positions of all **N** molecules, *B* is 1/(k_B T), k_B is Boltzmann's constant, and U(**r**) is the potential energy of the system. This integral is not analytic either (in fact the denominator is the same integral as discussed above) but numerical integration now yields the desired thermodynamic properties.

Numerical integration works by evaluating the integrand at many points inside the integration region and using the average value to estimate the integral. The accuracy of numeric integration is highest when you evaluate at a large number of points and when the function does not change rapidly over short distances. Neither of these are the case for Equation 1 because this is a 3**N** dimensional integral and if any two particles are overlapping then they have a very large, positive potential energy that leads to a zero in the exponential. Since we are integrating over all of the positions, then even for a system of only 10 particles we would have 30 degrees of freedom (x, y, and z coordinates for each particle). Breaking each of these up into just 10 points to evaluate per degree of freedom would result in 10^{30} computations. My old Pentium computer could do about 50 million computations per second so it would take about 10^{14} years to complete this calculation. As this is appreciably longer than the 10^{10} years the universe has existed it is clearly not a viable option without an amazing improvement in computational power.

It is possible to take advantage of the very property that makes this a difficult numerical integration, namely the fact that even though the configurational phase space (the **3N** dimensional volume that we need to sample) is huge, most of it does not contribute anything significant to the integral. What we do instead is start the system in one of the "good" states (set of positions) and propagate it through either time or ensemble space in such a way that the fraction of time it spends in any particular state is given by

$$\rho_i = \frac{\exp[-\beta U(\mathbf{r}_i)]}{\int_{\mathbf{r}} \exp[-\beta U(\mathbf{r})] d\mathbf{r}} \qquad Eq.2$$

where $\boldsymbol{p}_\mathbf{i}$ is the probability that the system is in state **i**. This is the idea behind molecular simulation. There are two different ways in which we can sample a system according to this probability distribution in such a way that we compute the average value of our observable without wasting computer time on the states that are unimportant. The two different ways of moving from one state to the next are called Molecular Dynamics (time average) and Monte Carlo (ensemble average) and they form the two branches of molecular simulation. While they are distinctly different approaches, they are equivalent from the viewpoint of statistical mechanics because the second postulate of statistical mechanics states that time averages are equivalent to ensemble averages.

Why choose Monte Carlo?

Molecular dynamics (MD) and Monte Carlo (MC) are equivalent methods, but they have different strengths and weaknesses. Molecular dynamics follows the natural time evolution of the system and this allows the calculation of time-dependent quantities like diffusion constants and viscosities. In MD you calculate the current forces on the system, compute the instantaneous velocities that would result from those forces, and assume that the molecules move with that velocity for a small increment of time. This "time step" typically ranges from around 0.5 to 10 fs, and this limits MD simulations to time scales under a microsecond. Monte Carlo does not follow the time evolution so dynamical quantities cannot be computed, but this also means that processes which take a long physical time can be studied if the simulation is designed properly. The Monte Carlo method also enables the use of certain ensembles specifically designed for computing phase equilibria (particularly the Gibbs ensemble) that are very difficult to simulate using molecular dynamics.

The phrase "Monte Carlo simulation" is used in a wide variety of contexts throughout the scientific literature. The main idea of all Monte Carlo simulations is to generate a large set of configurations and to measure the average (and sometimes variance) of some quantity of the system. These simulations are named after the famous gambling location Monte Carlo due to the random numbers that are used in order to generate and accept or reject trial moves. In our case, a Monte Carlo simulation is used in order to sample configurations according to a statistical mechanical ensemble.

The main algorithmic challenge of designing a Monte Carlo molecular simulation lies in devising ways to adequately and efficiently sample the equilibrium distribution of the correct statistical mechanical ensemble. If we can devise an algorithm that samples states with the probability distribution given in Equation 2 then we will be able to compute the canonical ensemble averages. Metropolis *et al.* were the first to show that you can sample such a distribution by treating the problem as if it were a Markov chain. A Markov chain is a collection of states where the probability of moving from one state to another depends only upon the state that the system is currently residing in, independent of how the system got into that current state. The trick is to select the probabilities of moving from one state to another in such a way that the system converges to a stationary distribution with the probabilities given in Equation 2.

This is best illustrated by considering a system of monatomic molecules in a periodic box. A periodic box is one in which molecules that exit out of one side of the box re-enter on the other side, and this is used to eliminate the effects of placing walls around the system. A Monte Carlo move is an attempt to change the system from one state to another. In this system the only move that is required to equilibrate the system (reach the stationary distribution) is a translational move. The algorithm for a translational move is as follows.
1) Select a molecule in the system at random.
2) Displace that molecule in a random direction by a distance $\mathbf{Z_1}$***M** where $\mathbf{Z_1}$ is a random number uniformly distributed over the interval (0,1), and **M** is the maximum displacement.
3) Compute the potential energy change [$\mathbf{U(r_{new}) - U(r_{old})}$] caused by moving this particle from its old location to the new location.
4) Accept or reject the move according to the acceptance probability

$$A_{\text{old}\rightarrow\text{new}} = \min[1, \exp(-\beta[U(r_{\text{new}}) - U(r_{\text{old}})])] \qquad Eq.3$$

If the energy change is negative, then the exponential term is greater than 1 and the move is accepted with probability 1. If the energy change is positive, then the move is accepted with probability $\mathbf{exp(-\mathit{B}\ [U(r_{new}) - U(r_{old})]\ )}$. This is done by computing a random number $\mathbf{Z_2}$ that is uniformly distributed over the interval (0,1). If $\mathbf{Z_2}$ is less than $\mathbf{exp(-\mathit{B}\ [U(r_{new}) - U(r_{old})]\ )}$, then the move is accepted and the new location is counted in the averaging, otherwise the molecule is returned to its original location and the old configuration is counted again in the averaging.

The reason for using this particular acceptance probability is revealed by analyzing the move according to the detailed balance condition in order to determine the stationary distribution of the Markov chain. For any two states in the system (**i** and **j**) the following must be true of the stationary distribution

$$\rho_i \times T_{i\rightarrow j} \times A_{i\rightarrow j} = \rho_j \times T_{j\rightarrow i} \times A_{j\rightarrow i} \qquad Eq.4$$

where $\boldsymbol{\rho_x}$ is the stationary probability the Markov chain is in state **x**, $\mathbf{T_{x \dashrightarrow y}}$ is the transition probability of attempting a move from state **x** to state **y**, and $\mathbf{A_{x \dashrightarrow y}}$ is the probability of accepting an attempted move from state **x** to state **y**. Let the potential energy of state **i** be higher than the potential energy of state **j**, (i.e. $\mathbf{U(r_i) > U(r_j)}$). We can then combine Equation 3 and Equation 4 and rewrite as

$$\frac{\rho_i}{\rho_j} = \frac{T_{j\to i} \times A_{j\to i}}{T_{i\to j} \times A_{i\to j}} = \frac{\exp\{-\beta[U(\mathbf{r}_i) - U(\mathbf{r}_j)]\}}{1} = \frac{\exp[-\beta U(\mathbf{r}_i)]}{\exp[-\beta U(\mathbf{r}_j)]} \qquad Eq.5$$

where the transition probabilities cancel out because the underlying Markov chain transition matrix is symmetric, that is $\mathbf{T}_{\mathbf{j} \text{-->} \mathbf{i}} = \mathbf{T}_{\mathbf{i} \text{-->} \mathbf{j}}$ for all **i** and **j**. We can also compute this ratio using Equation 2.

$$\frac{\rho_i}{\rho_j} = \frac{\dfrac{\exp[-\beta U(\mathbf{r}_i)]}{\int_{\mathbf{r}} \exp[-\beta U(\mathbf{r})]d\mathbf{r}}}{\dfrac{\exp[-\beta U(\mathbf{r}_j)]}{\int_{\mathbf{r}} \exp[-\beta U(\mathbf{r})]d\mathbf{r}}} = \frac{\exp[-\beta U(\mathbf{r}_i)]}{\exp[-\beta U(\mathbf{r}_j)]} \qquad Eq.6$$

Since these two quantities are equal, this demonstrates that the stationary distribution of this Markov chain is identical to the canonical distribution. This proof needs to be performed whenever a new move is proposed in order to insure that you are sampling from the desired statistical mechanical ensemble.

The next step

In theory, the Metropolis translation move is sufficient to sample the canonical ensemble. In practice, many other different kinds of moves are also utilized in order to reduce the amount of computer time required to get good convergence to the stationary distribution, and also to sample ensembles other than canonical. The broadly stated goal of Monte Carlo algorithm development is to achieve the best statistical precision using the least amount of computer time. Biased Monte Carlo methods is an active area of research and the Configurational-bias Monte Carlo page explains how these methods enable efficient simulation of molecules with complex architectures (long, branched and cyclic molecules) by utilizing an asymmetric underlying Markov chain transition matrix that makes it more likely to attempt to move to a molecular conformation with a lower energy than to attempt to move to one with a higher energy.

While algorithm power controls the precision of the simulation, the intermolecular potential functions (also known as force fields) control the accuracy of a simulation that is attempting to reproduce the behavior of real molecules. More information about the force fields implemented into Towhee is found on the Towhee Capabilities page

---

Send comments to: Marcus G. Martin

Last updated: December 14, 2006

MCCCS Towhee (Related Software)

Publicly Available Monte Carlo Molecular Simulation Packages

- BOSS: Bill Jorgensen's Monte Carlo package for simulations in single simulation box ensembles.
- Etomica: David Kofke's Java based Monte Carlo package originally designed for use as a teaching tool, but potentially powerful enough for research grade application.
- MedeA-GIBBS: Alain Fuch's Gibbs ensemble Monte Carlo package that contains many of the same algorithms implemented into Towhee, plus the use of anisotropic united-atoms. I am unclear on how exactly to acquire this package, but believe the folks at Materials Design are currently providing it to select customers.

Programs that Towhee can use as Libraries

- SeqQuest: Peter Schultz's package for local orbital Quantum Density Functional Theory calculations. This can be linked into Towhee to allow the use of Quantum Mechanical energies during a Monte Carlo simulation.
- Tramonto/FasTram: Laura Frink's package for Classical Density Functional Theory calculations. This can be linked into Towhee and used as an implicit solvation potential.

Useful Programs in combination with Towhee

- The Vimes GUI for Towhee: The Vimes project has created a GUI for Towhee that is currently in a beta tester stage. If you would like to give it a try please let them (and us) know how it works for your needs.
- Grace: graphing program that works with the examples and scripts in Towhee.
- LAMMPS: molecular dynamics simulator that Towhee can read and write output for to allow conversion between MD and MC methods.
- Rasmol: program for viewing pdb files.

Return to the main towhee web page

Send comments to: Marcus G. Martin

Last updated: December 14, 2006

MCCCS Towhee Force Fields

MCCCS Towhee (Weiner et al 1984)

Overview

This section covers an early force field in the Amber family called the Weiner *et al.* 1984 force field as it is implemented into the Towhee code. All of the Towhee atom types for the Weiner *et al.* 1984 force field are listed, along with a short description of their meanings. For more information about the Amber family of force fields see the Amber web site. Weiner *et al.* 1984 is a united-atom force field that was later expanded, and slightly modified, to allow some all atom types along side of the united atom types. See Weiner *et al.* 1986 for more information about this extension. Note that Weiner *et al.* 1984 is a 12-6 plus 12-10 H-bond force field and can only be combined with other '12-6 plus 12-10 H-bond' force fields. Any discrepencies (especially typos) from the published force field values are the sole responsibility of Marcus G. Martin, and I welcome feedback on how this implementation compares with other programs.

References for Weiner *et al.* 1984

- Weiner *et al.* 1984.

Weiner *et al.* 1984 in Towhee

The official force field name for Weiner *et al.* in Towhee is 'Weiner1984' and the file name is towhee_ff_Weiner1984 in the ForceFields directory. Here I list all of the atom names for use in the towhee_input file, along with a brief description taken from the Weiner *et al.* 1984 paper. Please note that the capitalization and spacing pattern is important and must be followed exactly as listed here.

Carbon (united or atomic)

- **'C2'** : united sp^3 carbon with two hydrogens
- **'C3'** : united sp^3 carbon with three hydrogens
- **'CD'** : united sp^2 aromatic carbon in six-membered ring with one hydrogen
- **'CE'** : united sp^2 aromatic carbon in five-membered ring between two nitrogens with one hydrogen (in purines)
- **'CF'** : united sp^2 aromatic carbon in five-membered ring next to a nitrogen without a hydrogen (e.g., C_{delta}-$N_{epsilon}$=$C_{epsilon}$ in histidine)
- **'CG'** : united sp^2 aromatic carbon in five membered ring next to a N-H (e.g., C_{delta}-$N_{epsilon}$=$C_{epsilon}$ in histidine)
- **'CH'** : united sp^3 carbon with one hydrogen
- **'CI'** : united sp^2 carbon in six-membered ring of purines between two "NC" nitrogens
- **'CJ'** : united sp^2 carbon in pyrimidines at positions 5 and 6 (more pure double bond than aromatic) with one hydrogen
- **'CP'** : united sp^2 aromatic carbon in five-membered ring between two nitrogens with one hydrogen (in His)
- **'C'** : atomic sp^2 carbonyl carbon and aromatic carbon with hydroxyl substituent in tyrosine
- **'C*'** : atomic sp^2 aromatic carbon in five-membered ring with one substituent (e.g., CE_{gamma} in Trp)
- **'CA'** : atomic sp^2 aromatic carbon in six-membered ring with one substituent
- **'CB'** : atomic sp^2 aromatic carbon at junction between five- and six-membered rings

(e.g., CE_{delta} in Trp, C4 and C5 in purines)

- **'CC'** : atomic sp^2 aromatic carbon in five-membered ring with one substituent and next to a nitrogen group (e.g., C_{gamma} in His)
- **'CM'** : atomic sp^2 same as CJ but one substituent
- **'CN'** : atomic sp^2 aromatic junction carbon in between five- and six-membered rings (e.g., $C_{epsilon}$ in Trp)
- **'CT'** : atomic sp^3 carbon with four explicit substituents

Hydrogen (all atomic)

- **'H3'** : hydrogens of lysine and arginine (positively charged)
- **'H2'** : amino hydrogens from NH_2 in purines and pyrimidines
- **'HC'** : explicit hydrogen attached to carbon
- **'H'** : amide and imino hydrogens
- **'HO'** : hydrogen on hydroxyl or water oxygen
- **'HS'** : hydrogen attached to sulfur

Nitrogen (all atomic)

- **'NC'** : sp^2 nitrogen in six-membered ring with lone pairs (e.g., N3 in adenine)
- **'NA'** : sp^2 nitrogen in five-membered ring with hydrogen attached (e.g., protonated His)
- **'NB'** : sp^2 nitrogen in five-membered ring with lone pairs (e.g., N7 in purines)
- **'N*'** : sp^2 nitrogen in purines and pyrimidines with alkyl group attached (N9 in purines. N1 in pyrimidines)
- **'N'** : sp^2 nitrogen in amide groups
- **'N2'** : sp^2 nitrogen in base NH_2 groups and arginines NH_2
- **'N3'** : sp^3 nitrogen with four substituents (e.g., Lys N_{zeta}
- **'NT'** : sp^3 nitrogen with three substituents (e.g., unprotonated amines)

Oxygen (all atomic)

- **'O'** : carbonyl oxygen
- **'O2'** : corboxyl and phosphate nonbonded oxygens
- **'OS'** : ether and ester oxygens
- **'OH'** : alcohol oxygens

Phosphorus (all atomic)

- **'P'** : phosphorus in phosphate groups

Sulfur (all atomic)

- **'S'** : sulfurs in disulfide linkages and mentioning
- **'SH'** : sulfur in cycstine
- **'LP'** : lone pairs attached to sulfur

Coulombic Interactions

Weiner *et al.* 1984 uses point charges located at atomic centers to represent the elecrostatic interactions between atoms. There is no simple, general table of charges for atoms in this force field. Instead, they recommend using the RESP method to fit the charges for each molecule of interest. See the Amber web site for more information about this method.

Improper torsions

Improper torsions are not automatically generated by the Towhee code as the rules for determining where they are applied are not always straight-forward. Weiner *et al.* exclusively uses the Stereocenter version of the improper torsions, and they are typically centered on an sp^2 atom in order to enforce planarity with its three neighbors. However, they are also centered on some united-atoms in order to enforce stereochemistry. Only one improper torsion allowed to be centered on any atom. These torsions are listed in the Amber literature as i-j-k-l where the angle is the dihedral between i-k-l and j-k-l, and the bonding pattern is i, j, and l are all bonded to atom k, and are also not bonded to each other. In the towhee_input file this stereo improper torsion is listed only for atom k, and the atom order there is l, i, j. Remember that you can set the improper type to 0 to have the code automatically determine the improper type (so long as inpstyle is 2). Please see the Weiner *et al.* 1984 paper for more information about assigning improper torsions for this force field.

Proteins

This force field contains parameters for all of the amino acids, but these have not yet been implemented into the polymer builder in Towhee. If anyone is interested in implementing this functionality for this force field please contact the Towhee developers.

Return to the Towhee Capabilities web page

Send comments to: Marcus G. Martin
Last updated: September 07, 2005

MCCCS Towhee (Weiner et al 1986)

Overview

This section covers an early force field in the Amber family called the Weiner *et al.* 1986 force field as it is implemented into the Towhee code. All of the Towhee atom types for the Weiner *et al.* 1986 force field are listed, along with a short description of their meanings. For more information about the Amber family of force fields see the Amber web site. Weiner *et al.* 1986 is an extension of a a previous force field called Weiner *et al.* 1984 that contained a substantial subset of the atom types available in Weiner *et al.* 1986, although please note that there were several small changes in parameters between 1984 and 1986. Additional parameters for Weiner *et al.* 1986 were later added for pentaerythritrol tetranitrate as decribed in the Dick and Ritchie 1994 manual. Note that Weiner *et al.* 1986 is a 12-6 plus 12-10 H-bond force field and can only be combined with other '12-6 plus 12-10 H-bond' force fields. Any discrepencies (especially typos) from the published force field values are the sole responsibility of Marcus G. Martin, and I welcome feedback on how this implementation compares with other programs.

References for Weiner *et al.* 1986

- Weiner *et al.* 1986.

Weiner *et al.* 1986 in Towhee

The official force field name for Weiner *et al.* in Towhee is 'Weiner1986' and the file name is towhee_ff_Weiner1986 in the ForceFields directory. Here I list all of the atom names for use in the towhee_input file, along with a brief description taken from the Weiner *et al.* 1984 and Weiner *et al.* 1986 papers. Please note that the capitalization and spacing pattern is important and must be followed exactly as listed here.

Carbon (united or atomic)

- **'C2'** : united sp^3 carbon with two hydrogens
- **'C3'** : united sp^3 carbon with three hydrogens
- **'CD'** : united sp^2 aromatic carbon in six-membered ring with one hydrogen
- **'CE'** : united sp^2 aromatic carbon in five-membered ring between two nitrogens with one hydrogen (in purines)
- **'CF'** : united sp^2 aromatic carbon in five-membered ring next to a nitrogen without a hydrogen (e.g., C_{delta}-$N_{epsilon}$=$C_{epsilon}$ in histidine)
- **'CG'** : united sp^2 aromatic carbon in five membered ring next to a N-H (e.g., C_{delta}-$N_{epsilon}$=$C_{epsilon}$ in histidine)
- **'CH'** : united sp^3 carbon with one hydrogen
- **'CI'** : united sp^2 carbon in six-membered ring of purines between two "NC" nitrogens
- **'CJ'** : united sp^2 carbon in pyrimidines at positions 5 and 6 (more pure double bond than aromatic) with one hydrogen
- **'CP'** : united sp^2 aromatic carbon in five-membered ring between two nitrogens with one hydrogen (in His)
- **'C'** : atomic sp^2 carbonyl carbon and aromatic carbon with hydroxyl substituent in tyrosine
- **'C*'** : atomic sp^2 aromatic carbon in five-membered ring with one substituent (e.g., CE_{gamma} in Trp)

2

'CA' : atomic sp aromatic carbon in six-membered ring with one substituent

- **'CB'** : atomic sp^2 aromatic carbon at junction between five- and six-membered rings (e.g., CE$_{delta}$ in Trp, C4 and C5 in purines)
- **'CC'** : atomic sp^2 aromatic carbon in five-membered ring with one substituent and next to a nitrogen group (e.g., C$_{gamma}$ in His)
- **'CK'** : atomic sp^2 aromatic carbon in five-membered ring between two nitrogens and bonded to one hydrogen (in purines)
- **'CM'** : atomic sp^2 same as CJ but one substituent
- **'CN'** : atomic sp^2 aromatic junction carbon in between five- and six-membered rings (e.g., C$_{epsilon}$ in Trp)
- **'CQ'** : atomic sp^2 carbon in six membered ring of purines between two "NC" nitrogens and bonded to one hydrogen.
- **'CR'** : atomic sp^2 aromatic carbon in five membered ring between two nitrogens and bonded to one hydrogen (in HIS)
- **'CT'** : atomic sp^3 carbon with four explicit substituents
- **'CV'** : atomic sp^2 aromatic carbon in five membered ring bonded to an N: and bonded to an explicit hydrogen (e.g. C$_{delta}$-N$_{epsilon}$=C$_{epsilon}$ in HIS)
- **'CW'** : atomic sp^2 aromatic carbon in five membered ring bonded to an N-H and bonded to an explicit hydrogen (e.g. C$_{delta}$-N$_{epsilon}$-C$_{epsilon}$ in HIS).

Hydrogen (all atomic)

- **'H3'** : hydrogens of lysine and arginine (positively charged)
- **'H2'** : amino hydrogens from NH$_2$ in purines and pyrimidines
- **'HC'** : explicit hydrogen attached to carbon
- **'H'** : amide and imino hydrogens
- **'HO'** : hydrogen on hydroxyl or water oxygen
- **'HS'** : hydrogen attached to sulfur

Nitrogen (all atomic)

- **'NC'** : sp^2 nitrogen in six-membered ring with lone pairs (e.g., N3 in adenine)
- **'NA'** : sp^2 nitrogen in five-membered ring with hydrogen attached (e.g., protonated His)
- **'NB'** : sp^2 nitrogen in five-membered ring with lone pairs (e.g., N7 in purines)
- **'N*'** : sp^2 nitrogen in purines and pyrimidines with alkyl group attached (N9 in purines. N1 in pyrimidines)
- **'N'** : sp^2 nitrogen in amide groups
- **'N2'** : sp^2 nitrogen in base NH$_2$ groups and arginines NH$_2$
- **'N3'** : sp^3 nitrogen with four substituents (e.g., Lys N$_{zeta}$
- **'NT'** : sp^3 nitrogen with three substituents (e.g., unprotonated amines)

Oxygen (all atomic)

- **'O'** : carbonyl oxygen
- **'O2'** : corboxyl and phosphate nonbonded oxygens
- **'OS'** : ether and ester oxygens
- **'OH'** : alcohol oxygens

Phosphorus (all atomic)

- **'P'** : phosphorus in phosphate groups

Sulfur (all atomic)

- **'S'** : sulfurs in disulfide linkages and mentioning
- **'SH'** : sulfur in cycstine
- **'LP'** : lone pairs attached to sulfur

Coulombic Interactions

Weiner *et al.* 1986 uses point charges located at atomic centers to represent the elecrostatic interactions between atoms. There is no simple, general table of charges for atoms in this force field. Instead, they recommend using the RESP method to fit the charges for each molecule of interest. See the Amber web site for more information about this method.

Improper torsions

Improper torsions are not automatically generated by the Towhee code as the rules for determining where they are applied are not always straight-forward. Weiner *et al.* exclusively uses the Stereocenter version of the improper torsions, and they are typically centered on an sp^2 atom in order to enforce planarity with its three neighbors. However, they are also centered on some united-atoms in order to enforce stereochemistry. Only one improper torsion allowed to be centered on any atom. These torsions are listed in the Amber literature as i-j-k-l where the angle is the dihedral between i-k-l and j-k-l, and the bonding pattern is i, j, and l are all bonded to atom k, and are also not bonded to each other. In the towhee_input file this stereo improper torsion is listed only for atom k, and the atom order there is l, i, j. Remember that you can set the improper type to 0 to have the code automatically determine the improper type (so long as inpstyle is 2). Please see the Weiner *et al.* 1986 paper for more information about assigning improper torsions for this force field.

Proteins

This force field contains parameters for all of the amino acids, but these have not yet been implemented into the polymer builder in Towhee. If anyone is interested in implementing this functionality for this force field please contact the Towhee developers.

Return to the Towhee Capabilities web page

Send comments to: Marcus G. Martin
Last updated: September 07, 2005

MCCCS Towhee (Dick and Ritchie 1994)

Overview

This section covers an extension to the Weiner *et al.* 1986 force field. Unlike most force fields, Dick and Ritchie 1994 is only useful in combination with the Weiner *et al.* 1986 force field. All of the additional Towhee atom types for the Dick and Ritchie 1994 extension are listed , along with a short description of their meanings. For more information about the Amber family of force fields see the Amber web site. Note that Dick and Ritchie 1994 is a 12-6 plus 12-10 H-bond force field and can only be combined with other '12-6 plus 12-10 H-bond' force fields. Any discrepencies (especially typos) from the published force field values are the sole responsibility of Marcus G. Martin, and I welcome feedback on how this implementation compares with other programs.

References for Dick and Ritchie 1994

- Dick and Ritchie 1994.

Dick and Ritchie 1994 in Towhee

The official force field name for Dick and Ritchie 1996 in Towhee is 'Weiner1986'. This indicates that Dick and Ritchie 1994 is not a stand alone force field and is only useful in combination with the Weiner *et al.* 1986 force field. In order to use the Dick and Ritchie 1994 extensions you need to include the towhee_ff_Dick1994 file from the ForceFields directory. Here I list all of the atom names for use in the towhee_input file, along with a brief description taken from Dick and Ritchie 1994. These atom types are designed to be mixed freely with the atom types described in the Weiner *et al.* 1986 documentation. Please note that the capitalization and spacing pattern is important and must be followed exactly as listed here.

Nitrogen (atomic)

- **'NN'** : sp^2 nitrogen in a nitro group.

Coulombic Interactions

Point charges for pentaerythritol tetranitrate (PETN) are provided in Table III of the Dick and Ritchie 1994 paper. However, the charges listed there add up to -0.003 for the PETN molecule so there must be some sort of typo in that table as the molecule should be net neutral. They also list different charges for the two symmetric oxygens on the nitro group, which might make sense for crystals, but does not make sense for fluids. Here I list their suggested charges for PETN and also the modifications I am currently using.

- Atom: Dick and Ritchie 1996 charges [Martin unpublished charges]
- CT (central carbon): -0.479 [-0.476]
- CT (branch carbon): -0.030 [-0.030]
- OS (ether-like oxygen): -0.476 [-0.476]
- NN (nitro nitrogen): 0.771 [0.771]
- O (nitro oxygens): -0.346 and -0.388 [-0.367]

Improper torsions

Improper torsions are not automatically generated by the Towhee code as the rules for determining where they are applied are not always straight-forward. Dick and Ritchie do not specify any additional improper torsion parameters so I do not believe any apply to the PETN molecule. For more information about improper torsions on other molecules please consult the

Weiner *et al.* 1986 web manual.

Return to the Towhee Capabilities web page

MCCCS Towhee (C19eef1)

Overview

This section covers the C19eef1 force field (which is a modified version of Charmm19 that includes the EEF1 implicit solvent model) as it is implemented into the towhee_ff_c19eef1 file in the ForceFields directory. All of the Towhee atom types for the C19eef1 force field are listed, along with a short description of their meanings. Note that Charmm19 is a '12-6 plus solvation' classical_potential with 'explicit' classical_mixrule so it cannot currently be combined with any other forcefields. The implicit water solvation is toggled on and off using the **isolvtype** variable in the towhee_input file. Any discrepencies (especially typos) from the published C19eef1 force field values are the sole responsibility of Marcus G. Martin, and I welcome feedback on how this implementation compares with other programs.

References for C19eef1

The best literature reference for the C19eef1 forcefield parameters is

- Neria *et al.* 1996

The best literature reference for the C19eef1 solvation parameters is

- Lazaridis and Karplus 1999

There are parameter (param19_eef1.inp) and topology (toph19_eef1.inp) files which are extremely helpful, but are only available in the Charmm28 distribution.

C19eef1 in Towhee

The official force field name for C19eef1 in Towhee is 'C19eef1'. Here I list all of the C19eef1 atom names for use in the towhee_input file, along with a brief description. C19eef1 denotes extended-atom for cases where the hydrogens and the heavy element they are bonded to are all lumped into a single interaction site. Please note that the capitalization and spacing pattern is important and must be followed exactly as listed here. I include the atom notes from the Neria paper.

- **'C'** : Carbonyl carbon
- **'CH1E'** : Extended aliphatic carbon with one hydrogen -CH- group
- **'CH2E'** : Extended aliphatic carbon with two hydrogens -CH_2- group
- **'CH3E'** : Extended methyl terminal -CH_3 group
- **'CM'** : Carbon in carbon monoxide
- **'CR'** : 4-bonded carbon in aromatics and Arginine
- **'CR1E'** : Extended aromatic carbon with one hydrogen
- **'CT'** : aliphatic carbon
- **'FE'** : Iron
- **'H'** : Polar hydrogen
- **'HC'** : Polar hydrogen (in Arg, Lys, and N term)
- **'HT'** : Water hydrogen, modified TIP3P model
- **'N'** : Nitrogen with no hydrogens
- **'NC2'** : Nitrogen bound to two hydrogens (in Arg.)
- **'NH1'** : Nitrogen bound to one hydrogen
- **'NH2'** : Nitrogen bound to two hydrogens
- **'NH3'** : Nitrogen bound to three hydrogens
- **'NP'** : Pyrole nitrogen
- **'NR'** : Nitrogen in an aromatic ring with no hydrogens
- **'O'** : Carbonyl oxygen
- **'OC'** : Carboxyl oxygen

- **'OH1'** : Hydroxyl oxygen
- **'OH2'** : ST2 water oxygen
- **'OM'** : Heme CO/O2 oxygen
- **'OS'** : Ester oxygen
- **'OT'** : Water oxygen, modified TIP3P model
- **'S'** : Sulfur
- **'SH1E'** : Extended sulfur with one hydrogen

Coulombic Interactions

C19eef1 utilizes point charges on atomic centers to represent the charge distribution on a molecule. As far as I know, there is no automated system for assigning the charges in C19eef1. However, C19eef1 uses a neutral group approach for most moities found in organic molecules. The charge distibution is similiar to that used in Charmm19, except the ionic side chain amino acids are neutralized, and the specifics on how to do this can be found in Lazaridis and Karplus 1999. Otherwise, it is fairly easy to scan through the example molecular charge distributions found in the files available at the MacKerell research web site and determine what charges to apply to the molecule you wish to simulate.

Improper torsions

Improper torsions are not automatically generated by the Towhee code as the rules for determining where they are applied are not always straight-forward. C19eef1 exclusively uses the out-of-plane version of the improper torsions, and they are typically centered on an sp^2 atom in order to enforce planarity with its three neighbors. These torsions are listed in the C19eef1 literature as i-j-k-l where the angle is the dihedral between i-j-k and j-k-l. The bonding pattern is not completely clear to me, but it appears that either i or l is the central atom which is bonded to all three of the other atoms, and none of the other three atoms are bonded to each other. In the towhee_input file the improper torsions is listed starting from the central atom and the three other atoms are listed in the same order as C19eef1. So, if the central atom is i, then the atoms are listed j, k, l. If the central atom is l then the atoms are listed k, j, i. Remember that you can set the improper type to 0 to have the code automatically determine the improper type (so long as inpstyle is 2).

Proteins

All of the 20 basic amino acids (including the 3 forms of hystidine) are functional for the C19eef1 force field. I have implemented the atom types and charges according to the published Ch19eef1 values. Below is a complete list of the codes for the 20 amino acids, plus some other functional groups that work with the protein builder. C19eef1 does not apply a torsion across all set of atoms connected by 3 bonds in the amino acids. The torsions are explicitly listed for these systems in the polyc19eef1.F routine. I tried to faithfully reproduce all of their energetics, but had to make some accomodations for tryptophan. They have two torsions in tryptophan that span atoms that are not bonded together and I could not implement those into Towhee without a major reworking of configuartional-bias.

- 'a0' alanine
- 'c0' cysteine with hydrogen on the sulfur
- 'd-' aspartic acid deprotonated
- 'e-' glutamic acid deprotonated
- 'f0' phenylalanine
- 'g0' glycine
- 'h+' histidine both N protonated
- 'hd' histidine neutral with only N_d protonated
- 'he' histidine neutral with only N_e protonated

- 'i0' isoleucine
- 'k+' lysine protonated
- 'l0' leucine
- 'm0' methionine
- 'n0' asparagine
- 'p0' proline (no parameters for N-terminal or C-terminal proline)
- 'q0' glutamine
- 'r+' arginine protonated
- 's0' serine
- 't0' threonine
- 'v0' valine
- 'w0' tryptophan
- 'y0' tyrosine

Return to the Towhee Capabilities web page

Send comments to: Marcus G. Martin
Last updated: May 12, 2006

MCCCS Towhee (C19sasa)

Overview

This section covers the C19sasa force field (which is a modified version of Charmm19 that includes an implicit solvent model based on the solvent accessible surface area (SASA)) as it is implemented into the towhee_ff_c19sasa file in the ForceFields directory. All of the Towhee atom types for the C19sasa force field are listed, along with a short description of their meanings. Note that Charmm19 uses a '12-6 plus solvation' classical_potential and 'explicit' classical_mixrule so it cannot currently be combined with any other forcefields. The implicit water solvation is toggled on and off using the **isolvtype** variable in the towhee_input file. Any discrepencies (especially typos) from the published C19sasa force field values are the sole responsibility of Marcus G. Martin, and I welcome feedback on how this implementation compares with other programs.

References for C19sasa

The best literature reference for the C19sasa forcefield parameters is

- Neria *et al.* 1996

The best literature reference for the C19sasa solvation parameters is

- Lazaridis and Karplus 1999

This forcefield also uses parameter (param19_eef1.inp) and topology (toph19_eef1.inp) files which are extremely helpful, but are only available in the Charmm28 distribution.

C19sasa in Towhee

The official force field name for C19sasa in Towhee is 'C19sasa'. Here I list all of the C19sasa atom names for use in the towhee_input file, along with a brief description. C19sasa denotes extended-atom for cases where the hydrogens and the heavy element they are bonded to are all lumped into a single interaction site. Please note that the capitalization and spacing pattern is important and must be followed exactly as listed here. I include the atom notes from the Neria paper.

- **'C'** : Carbonyl carbon
- **'CH1E'** : Extended aliphatic carbon with one hydrogen -CH- group
- **'CH2E'** : Extended aliphatic carbon with two hydrogens -CH_2- group
- **'CH3E'** : Extended methyl terminal -CH_3 group
- **'CM'** : Carbon in carbon monoxide
- **'CR'** : 4-bonded carbon in aromatics and Arginine
- **'CR1E'** : Extended aromatic carbon with one hydrogen
- **'CT'** : aliphatic carbon
- **'FE'** : Iron
- **'H'** : Polar hydrogen
- **'HC'** : Polar hydrogen (in Arg, Lys, and N term)
- **'HT'** : Water hydrogen, modified TIP3P model
- **'N'** : Nitrogen with no hydrogens
- **'NC2'** : Nitrogen bound to two hydrogens (in Arg.)
- **'NH1'** : Nitrogen bound to one hydrogen
- **'NH2'** : Nitrogen bound to two hydrogens
- **'NH3'** : Nitrogen bound to three hydrogens
- **'NP'** : Pyrole nitrogen
- **'NR'** : Nitrogen in an aromatic ring with no hydrogens
- **'O'** : Carbonyl oxygen

- **'OC'** : Carboxyl oxygen
- **'OH1'** : Hydroxyl oxygen
- **'OH2'** : ST2 water oxygen
- **'OM'** : Heme CO/O2 oxygen
- **'OS'** : Ester oxygen
- **'OT'** : Water oxygen, modified TIP3P model
- **'S'** : Sulfur
- **'SH1E'** : Extended sulfur with one hydrogen

Coulombic Interactions

C19sasa utilizes point charges on atomic centers to represent the charge distribution on a molecule. As far as I know, there is no automated system for assigning the charges in C19sasa. However, C19sasa uses a neutral group approach for most moities found in organic molecules. The charge distibution is similiar to that used in Charmm19, except the ionic side chain amino acids are neutralized, and the specifics on how to do this can be found in Lazaridis and Karplus 1999. Otherwise, it is fairly easy to scan through the example molecular charge distributions found in the files available at the MacKerell research web site and determine what charges to apply to the molecule you wish to simulate.

Improper torsions

Improper torsions are not automatically generated by the Towhee code as the rules for determining where they are applied are not always straight-forward. C19sasa exclusively uses the out-of-plane version of the improper torsions, and they are typically centered on an sp^2 atom in order to enforce planarity with its three neighbors. These torsions are listed in the C19sasa literature as i-j-k-l where the angle is the dihedral between i-j-k and j-k-l. The bonding pattern is not completely clear to me, but it appears that either i or l is the central atom which is bonded to all three of the other atoms, and none of the other three atoms are bonded to each other. In the towhee_input file the improper torsions is listed starting from the central atom and the three other atoms are listed in the same order as C19sasa. So, if the central atom is i, then the atoms are listed j, k, l. If the central atom is l then the atoms are listed k, j, i. Remember that you can set the improper type to 0 to have the code automatically determine the improper type (so long as inpstyle is 2).

Proteins

All of the 20 basic amino acids (including the 3 forms of hystidine) are functional for the C19sasa force field. I have implemented the atom types and charges according to the published C19sasa values. Below is a complete list of the codes for the 20 amino acids, plus some other functional groups that work with the protein builder. C19sasa does not apply a torsion across all set of atoms connected by 3 bonds in the amino acids. The torsions are explicitly listed for these systems in the polyc19eef1.F routine. I tried to faithfully reproduce all of their energetics, but had to make some accomodations for tryptophan. They have two torsions in tryptophan that span atoms that are not bonded together and I could not implement those into Towhee without a major reworking of configuartional-bias.

- 'a0' alanine
- 'c0' cysteine with hydrogen on the sulfur
- 'd-' aspartic acid deprotonated
- 'e-' glutamic acid deprotonated
- 'f0' phenylalanine
- 'g0' glycine
- 'h+' histidine both N protonated
- 'hd' histidine neutral with only N_d protonated

- 'he' histidine neutral with only N_e protonated
- 'i0' isoleucine
- 'k+' lysine protonated
- 'l0' leucine
- 'm0' methionine
- 'n0' asparagine
- 'p0' proline (no parameters for N-terminal or C-terminal proline)
- 'q0' glutamine
- 'r+' arginine protonated
- 's0' serine
- 't0' threonine
- 'v0' valine
- 'w0' tryptophan
- 'y0' tyrosine

Return to the Towhee Capabilities web page

Send comments to: Marcus G. Martin
Last updated: May 12, 2006

MCCCS Towhee (Fris2003)

Overview

This section covers the Frischknecht and Curro United-Atom force field for poly(dimethylsiloxane) as it is implemented in the towhee_ff_Fris2003 file in the ForceFields directory. All of the Towhee atom types for this force field are listed, along with a short description of their meanings. Note that this force field is a hybrid of Lennard-Jones (12-6) and Lennard-Jones (9-6) which ends up being a 12-9-6 potential and their implementation does not have general mixing rules for combination with other potentials. I would like to acknowledge Amalie L. Frischknecht for providing very useful guidance about implementing this force field. Any discrepencies (especially typos) from the published force field values are the sole responsibility of Marcus G. Martin, and I welcome feedback on how this implementation compares with other programs.

References for Frischknecht and Curro (Fris2003)

This force field is a hybrid of an earlier version of the Compass force field and a united-atom forcefield of Sok *et al.*. Please see the following reference for more details.

- Frischknecht and Curro 2003

with the exception that there is a typo in Table 1 of that paper which states the units for the angle force constant are kcal/(mol deg^2). They are actually listed in the more typical kcal/(mol rad^2) (personal communication between M.G. Martin and A.L. Frischknecht October 2003).

Frischknecht and Curro in Towhee

The official force field name for these parameters is 'Fris2003' (starting with version 3.13.0) Here I list all of the atom names for use in the towhee_input file, along with a brief description of what atoms we are talking about. Please note that the capitalization and spacing pattern is important and must be followed exactly as listed here.

- **'CH3'** : united-atom methyl group (one carbon and 3 hydrogens)
- **'O'** : oxygen
- **'Si'** : silicon

Coulombic Interactions

This force field uses point charges on atomic centers. They offer several possible charge sets in their paper, but it looks like they recomend two possible models. In the first, the charge is set to 0.0 on every atom (that is, there are no coulombic interactions). In the second, the charge is still 0.0 on the methyl groups, Si has a 0.3 charge, and O has a -0.3 charge.

Improper torsions

There are no improper torsions for this force field.

Proteins

This force field does not have parameters for proteins.

Return to the Towhee Capabilities web page

Send comments to: Marcus G. Martin
Last updated: September 07, 2005

MCCCS Towhee (COMPASSv1)

Overview

This section covers the COMPASSv1 force field as it is implemented into the towhee_ff_COMPASSv1 file in the ForceFields directory. All of the Towhee atom types for the COMPASSv1 force field are listed, along with a short description of their meanings. For more information about the COMPASSv1 force field see the Accelrys web site. The COMPASSv1 force field is maintained by researchers at Accelrys and only a portion of it has been published in the peer-reviewed literature. I have only incorporated those parts that have been published. Note that COMPASSv1 is a Lennard-Jones (9-6) force field and can only be combined with other Lennard-Jones (9-6) force fields. I would like to acknowledge David Rigby for providing very useful guidance about implementing COMPASSv1. Any discrepencies (especially typos) from the published COMPASSv1 force field values are the sole responsibility of Marcus G. Martin, and I welcome feedback on how this implementation compares with other programs.

References for COMPASSv1

The COMPASS family of forcefield is an ongoing subject of further parameterization. The authors of the forcefield periodically revisit old parameters and refit them to additional data. Towhee defines COMPASSv1 as the parameters published in the following series of papers.

- Rigby *et al.* 1997
- Sun and Rigby 1997
- Sun 1998
- Sun *et al.* 1998
- Bunte and Sun 2000
- Yang *et al.* 2000
- McQuaid *et al.* 2003

Typos and clarifications for COMPASSv1

With the large size of the COMPASSv1 force field it is not surprising that there are a few typos in the published literature. Here I summarize and clarify differences in my implementation from the literature.

- There are full van der Waals and coulombic interactions between atoms connected by three bonds (1-4 interactions).
- Rigby *et al.* 1997: In the angle-angle-torsion term for ha-c4-c4-o2 the ijk/jkl/ijkl constant is set to 20.2006 while it should actually be -20.2006 [personal communication between M.G. Martin and D. Rigby August 09, 2001]
- Sun 1998: The mixing rules for epsilon are stated incorrectly. The term in the denominator should read $(r_i^0)^6 + (r_j^0)^6$. The + is missing in that paper.
- Sun 1998: In the End Bond-Torsion for c4-c4-c4-c4 the K-L/I-J-K-L k1 value is set to 0.0000 and I think it should be set to the same value as the I-J/I-J-K-L k1 value of -0.0732. This correction was made in Towhee.

COMPASSv1 in Towhee

The official force field name for COMPASSv1 in Towhee is 'COMPASSv1'. Here I list all of the COMPASSv1 atom names for use in the towhee_input file, along with a brief description taken from the COMPASSv1 papers. I have added some comments where I thought clarification was needed, these are all in [square brackets]. The COMPASSv1 naming convention has the first letter as the element, the second character is the number of bonds to that element, and the third character is a bit of a wildcard. Please note that the capitalization and spacing pattern is important and must be followed exactly as listed here.

Argon
- **'ar'** : argon

Carbon
- **'c1o'** : carbon in CO
- **'c2='** : carbon in CO_2 and CS_2
- **'c3a'** : aromatic carbon [carbon with 3 bonds, aromatic]
- **'c4'** : generic sp^3 carbon [carbon with 4 bonds, use only if none of the other c4 special cases are valid]
- **'c43'** : sp^3 carbon with three heavy atoms attached
- **'c44'** : sp^3 carbon with four heavy atoms attached
- **'c4o'** : alpha carbon [sp^3 carbon bonded to an oxygen in an alcohol or ether]
- **'c4z'** : carbon, sp^3, bonded to $-N_3$

Helium
- **'he'** : helium

Hydrogen
- **'h1'** : nonpolar hydrogen [hydrogen bonded to carbon]
- **'h1h'** : hydrogen in H_2
- **'h1o'** : strongly polar hydrogen [hydrogen bonded to oxygen]

Krypton
- **'kr'** : krypton

Neon
- **'ne'** : neon

Nitrogen
- **'n1n'** : nitrogen in N_2
- **'n1o'** : nitrogen in NO
- **'n1z'** : nitrogen, terminal atom in $-N_3$
- **'n2='** : nitrogen [nitrogen with a single bond and a double bond]
- **'n2o'** : nitrogen in NO_2
- **'n2z'** : nitrogen, first atom in $-N_3$
- **'n2t'** : nitrogen, central atom in $-N_3$
- **'n3o'** : nitrogen in nitro group

Oxygen
- **'o1='** : oxygen in NO_2 and SO_2
- **'o1=*'** : oxygen in CO_2
- **'o12'** : oxygen in nitro group ($-NO_2$)
- **'o1c'** : oxygen in CO
- **'o1n'** : oxygen in NO
- **'o1o'** : oxygen in O_2
- **'o2'** : generic oxygen with two bonds
- **'o2e'** : ether oxygen
- **'o2h'** : hydroxyl oxygen [alcohol oxygen]
- **'o2n'** : oxygen in nitrates

Phosphorous
- **'p4='** : phosphorous [phosphorous with three single bonds and a double bond]

Silicon
- **'si4'** : generic silicon with four bonds attached
- **'si4c'** : a subset of Si4, non-hydrogen atom attached

Sulfur

- **'s1='** : sulfur in CS_2
- **'s2='** : sulfur in SO_2
- **'xe'** : xenon

Coulombic Interactions

COMPASSv1 uses point charges to represent the electrostatic interactions on molecules. They have devised an automated system for assigning the point charges on each atom that depends on the atom types, and the types of atoms bonded to each atom. This system is implemented into Towhee Version 4.4.0 and later. In order to anable automatic charge assignment you use **inpstyle** 2 in combination with **charge_assignment** 'bond increment'. Please see the inpstyle 2 documentation for further information on enabling this option.

Angle-Angle cross terms

COMPASSv1 is a class 2 force field and so you do need to include angle-angle cross terms. However, I have automated the detection and type determination for COMPASSv1 so I would recomend using the inpstyle = 2 feature in order to determine these. The rest of the fancy cross terms for COMPASSv1 are included in the bending angle and dihedral angle potentials.

Improper torsions

Improper torsions are not automatically generated by the Towhee code as the rules for determining where they are applied are not always straight-forward. COMPASSv1 exclusively uses the out-of-plane version of the improper torsions, and they are typically centered on an sp^2 atom in order to enforce planarity with its three neighbors. These torsions are listed in the COMPASSv1 literature as i-j-k-l where the angle is the dihedral between j-i-k and i-k-l, and the bonding pattern is i, k, and l all bonded to atom j, and also not bonded to each other. In the towhee_input file this out-of-plane improper torsion is listed only for atom j, and the atom order there is i, k, l. Unfortunately, I have not yet set up the automation for determining the improper types for COMPASSv1 yet as the impropers are almost exclusively used on aromatics, and I have not yet tested the code for those molecules. You will need to look in fftorsion.F in order to find the types if you need the improper torsions for COMPASSv1.

Return to the Towhee Capabilities web page

Send comments to: Marcus G. Martin
Last updated: September 07, 2005

MCCCS Towhee (QMFF-VIII)

Overview

This section covers the QMFF-VIII force field as it is implemented into the towhee_ff_QMFF-VIII file in the ForceFields directory. This force field was designed for a variety of organic liquids. Note that this is a Class-2 9-6 force field that uses the 'Sixth Power' mixing rules. Any discrepencies (especially typos) from the published values are the sole responsibility of Marcus G. Martin, and I welcome feedback on how this implementation compares with other programs.

References for QMFF-VIII

This forcefield is described in a single paper, and the supplementary information for that paper.

- Ewig *et al.* 2001

Typos and comments for QMFF-VIII

- The supplementary information contains a few additional parameters that were not mentioned in the main paper. These have been implemented, although I am not positive about how they were intended to be used.
- The supplementary information does not contain the c= h* and n= parameters. I believe that a formatting error has replaced those symbols with additional c h and n parameters because there are duplicates of those listed there. I have been unable to get a useful response from any of the authors and therefore have used some logic combined with inspiration in order to attempt to reproduce the true values from the duplicates. Here I list all of the changes I made to the supplementary information in order to create the values implemented into Towhee.
    - The **h** and **h*** atom types are distinct, but given that they are mixed together due to a formatting error in the file, and that they are bonded to different atom types, all of their bonded terms are treated as a generic **h**. This will work properly so long as the **h** atom types are only bonded to carbon and sulfur while the **h*** atom types are only bonded to oxygen and nitrogen.
    - The **c=** term was erroneously listed as an additional **c** term in the supplementary information. Any time a duplicate term is listed that includes a **c** I assume that this was actually supposed to be a **c=**. Given that most of the terms are listed in order in that file it is relatively easy to figure out which terms should be **c=**.
    - The **n=** term was erroneously listed as an additional **n** term in the supplementary information. Any time a duplicate term is listed that includes a **n** I assume that this was actually supposed to be a **n=**. Given that most of the terms are listed in order in that file it is relatively easy to figure out which terms should be **n=**.
    - An additional oxygen term is listed and I do not know the purpose for which it was originally intended. I have labeled the second of any duplicate oxygens as **o?**.
    - Page 50 Line 29: assumed that the duplicate van der Waals listing for **c** was intended for **c=**
    - Page 50 Line 38: assumed that the duplicate van der Waals listing for **h** was intended for **h***
    - Page 50 Line 44: assumed that the duplicate van der Waals listing for **n** was intedned for **n=**
    - Page 51 Line 5: a duplicate van der Waals listing for **o** is on this line is a mystery and was set to the mystery oxygen **o?**

QMFF-VIII in Towhee

The official force field name for QMFF-VIII in Towhee is 'Shah2004'. Here is the list of atom names for use in the towhee_input file, along with a brief description. These names are from the main text of the paper, and also include atoms from the supplementary information that do not have a description of their proper use. Please note that the capitalization and spacing pattern is important and must be followed exactly as listed here.

Bromine

- **Br** : no explanation provided. This was assigned a mass appropriate for bromine and I suspect it is intended for Br^- ion.
- **br** : no explanation provided. This was assigned a mass appropriate for bromine and I suspect it is intended for a bonded bromine, but while bond increment parameters are included, there are no bond vibration parameters so it is not usable as currently implemented.

Calcium

- **Ca** : no explanation provided. This was assigned a mass appropriate for calcium and I suspect it is intended for Ca^{2+} ion.

Carbon

- **c** : carbon (sp^3) in alkanes
- **c"** : carbonyl carbon except in amides. Be sure that you are surrounding this atom type with single quotes in towhee_input as using double quotes will cause a read error for **c"**
- **c`** : carbonyl carbon in amides (connected to nitrogen). This is listed as **c'** in their work, but is implemented using the alternate style of single quote in order to avoid problems reading character strings surrounded by the typical Fortran single quotes.
- **c+** : triply connected carbon in amidine and imidazole cations.
- **c-** : carbon in carboxylate groups
- **c=** : carbon (sp^2) in alkenes. This atom type was erronously listed as **c** in the supplementary information so some inference was required in order to determine which parameters belong to **c=** instead of **c**. See the Typos and Comments section above for a complete list of inferred parameters for this atom type.
- **co** : anomeric carbon in acetals and hemiacetals.
- **cp** : carbon in aromatic rings.
- **cpb** : bridging carbon in bridged aromatics
- **cr** : no explanation provided. This was assigned a mass appropriate for carbon and I suspect it is some form of bonded carbon. The only other parameters provided for this type are bond increment values with a few forms of nitrogen so this is probably not useful as implemented into Towhee.
- **ct** : carbon (sp) in alkynes.

Chlorine

- **Cl** : no explanation provided. This was assigned a mass appropriate for chlorine and I suspect it is intended for Cl^- ion.
- **cl** : chlorine. This appears to be intended for use when bonded to other atoms.

Cesium

- **Cs** : no explanation provided. This was assigned a mass appropriate for cesium and I suspect it is intended for Cs^+ ion.

Copper

- **Cu** : no explanation provided. This was assigned a mass appropriate for copper and I suspect it is intended for a Cu ion, but I am not sure of the charge state.

Fluorine

- **F** : no explanation provided. This was assigned a mass appropriate for fluorine and I suspect it is intended for F^- ion.
- **f** : fluorine. This appears to be intended for use when bonded to other atoms.

Hydrogen

- **h** : hydrogen connected to carbon or sulfur.
- **h*** : hydrogen connected to oxygen or nitrogen (except in ammonium). This atom type

was erroneously listed as **h** in the supplementary information so some inference was required in order to determine which parameters belong to **h*** insteda of **h**. See the Typos and Comments section above for a complete list of inferred parameters for this atom type.

- **h+** : hydrogen in ammonium groups.

Iron

- **Fe** : no explanation provided. This was assigned a mass appropriate for iron and I suspect it is intended for an Fe ion, but I am not sure of the charge state.

Iodine

- **I** : no explanation provided. This was assigned a mass appropriate for iodine and I suspect it is intended for I^- ion.
- **i** : no explanation provided. This was assigned a mass appropriate for iodine and I suspect it is intended for iodine bonded to another atom. Unforunately the only other parameters provided for this atom type are bond increments so this is probably not useful as implemented into Towhee.

Potassium

- **K** : no explanation provided. This was assigned a mass appropriate for potassium and I suspect it is intended for K^+ ion.

Lithium

- **Li** : no explanation provided. This was assigned a mass appropriate for lithium and I suspect it is intended for Li^+ ion.

Magnesium

- **Mg** : no explanation provided. This was assigned a mass appropriate for magnesium and I suspect it is intended for Mg^{2+} ion.

Nitrogen

- **n** : nitrogen in amides and amidines
- **n`** : nitrogen in nitro groups. This is listed as **n'** in their work, but is implemented using the alternate style of single quote in order to avoid problems reading character strings surrounded by the typical Fortran single quotes.
- **n+** : nitrogen in ammonium groups.
- **n=** : doubly bonded nitrogen. This atom type was erronously listed as **n** in the supplementary information so some inference was required in order to determine which parameters belong to **n=** instead of **n**. See the Typos and Comments section above for a complete list of inferred parameters for this atom type.
- **na** : nitrogen (sp^3) in amines
- **nh** : nitrogen (sp^2) triply connected in five- and six-membered rings (e.g. pyrrole)
- **nn** : nitrogen (sp^2) connected to aromatic rings (e.g. aniline)
- **np** : nitrogen (sp^2) doubly connected in five- and six-membered rings (e.g. pyridine)

Oxygen

- **o** : doubly connected oxygen
- **o`** : carbonyl and sulfonyl oxygen. This is listed as **o'** in their work, but is implemented using the alternate style of single quote in order to avoid problems reading character strings surrounded by the typical Fortran single quotes.
- **o-** : oxygen in carboxylates, dialkyl phosphaes, and phosphate dianions
- **op** : oxygen in aromatic rings
- **o?** : no explanation provided. There are several instances of duplicate oxygen parameters and I suspect it is an additional unpublished oxygen value that got mixed up with the normal **o** in the formatting problems. I have given it the **o?** name and included it in case anyone knows its true purpose.

Phosphorus

- **p** : phosphorus in dialkyl phosphates
- **p-** : phosphorus in phosphate dianions

Sodium

- **Na** : no explanation provided. This was assigned a mass appropriate for sodium and I suspect it is intended for Na^+ ion.

Sulfur

- **s** : sulfur (sp^3) doubly connected
- **s"** : sulfur in solfones and solfonamides. Be sure that you are surrounding this atom type with single quotes in towhee_input as using double quotes will cause a read error for **s"**
- **sp** : sulfur in aromatic rings

Rubidium

- **Rb** : no explanation provided. This was assigned a mass appropriate for rubidium and I suspect it is intended for Mg^+ ion.

Zinc

- **Zn** : no explanation provided. This was assigned a mass appropriate for zinc and I suspect it is intended for a Zn ion, but am not sure of the charge state.

Coulombic interactions

This force field uses point charges and has been set up to assign the point charges using the bond increment method.

Return to the Towhee Capabilities web page

Send comments to: Marcus G. Martin
Last updated: December 11, 2006

MCCCS Towhee (MMFF94)

Overview

This section covers the MMFF94 force field as it is implemented into the towhee_ff_MMFF94 file in the ForceFields directory. This force field was parameterized for a wide variety of organic liquids. Note that this is a Class-2 'Buffered 14-7' force field that uses the 'MMFF' mixing rules. Any discrepencies (especially typos) from the published values are the sole responsibility of Marcus G. Martin, and I welcome feedback on how this implementation compares with other programs.

References for MMFF94

This forcefield is described in a series of papers, and the supplementary information for those papers.

- Halgren 1996 (I)
- Halgren 1996 (II)
- Halgren 1996 (III)
- Halgren and Nachbar 1996 (IV)
- Halgren 1996 (V)
- Halgren 1999 (VII)

Typos and comments for MMFF94

- The supplementary information is very useful and is available via anonymous ftp at **ftp.wiley.com** in the **/public/journals/jcc/suppmat/17/490** directory. Those files are the primary source for the implementation in Towhee.
- There is useful additional information about the definitions of the angle types. stretch-bend types, and torsion types available on the CHARMM MMFF94 params website. Note that there are two logic errors in the CHARMM MMFF94 stretch-bend
  - Their table lists a stretch-bend type of 2 for an angle type of 2, a bond(ij) type of 0 and a bond(jk) type of 1. This is impossible as an angle that is not in a three or four membered ring that has those bond types has an angle type of 1, not 2. Towhee assigns a stretch-bend type of 2 for an angle type of 1, a bond(ij) of 0, and a bond(jk) of 1.
  - Their table lists a stretch-bend type of 11 for an angle type of 7, a bond(ij) type of 1 and a bond(jk) type of 1. This is impossible as an angle that is in a four membered ring with has those bond types has an angle type of 8, not 7. Towhee assigns a stretch-bend type of 11 for an angle type of 8, a bond(ij) of 1, and a bond(jk) of 1.
- A useful test suite of compounds and MMFF energies is available at http://www.ccl.net/cca/data/MMFF94/. The Towee implementation has not yet been validated for these compounds (hopefully soon).

MMFF94 in Towhee

The official force field name for MMFF94 in Towhee is 'MMFF94'. This section provides a complete list of the implemented nonbonded parameter names for use in the towhee_input file, along with a brief description. The names and descriptions are taken from the MMFFSYMB.PAR section of appendix B in the supplementary information file. The (MMFF number) included in the description is the integer code used in the internals of the MMFF force field. If two atom types have the same number then they have the same interactions. That means there are redundant atom types in the MMFF force field. These names and descriptions are from Table III of Halgren 1996 (I). Note that the capitalization and spacing pattern is important and must be followed exactly as listed here.

Bromine

BR : (MMFF number 13) BROMINE
- **BR-** : (MMFF number 91) BROMIDE ANION

Calcium
- **CA+2** : (MMFF number 96) DIPOSITIVE CALCIUM

Carbon
- **CR** : (MMFF number 1) ALKYL CARBON, SP3
- **C=C** : (MMFF number 2) VINYLIC CARBON, SP2
- **CSP2** : (MMFF number 2) GENERIC SP2 CARBON
- **C=O** : (MMFF number 3) GENERAL CARBONYL CARBON
- **C=N** : (MMFF number 3) SP2 CARBON IN C=N
- **CGD** : (MMFF number 3) GUANIDINE CARBON, DOUBLY BONDED TO N
- **C=OR** : (MMFF number 3) KETONE OR ALDEHYDE CARBONYL CARBON
- **C=ON** : (MMFF number 3) AMIDE CARBONYL CARBON
- **CONN** : (MMFF number 3) UREA CARBONYL CARBON
- **COO** : (MMFF number 3) CARBOXYLIC ACID OR ESTER CARBONYL CARBON
- **COON** : (MMFF number 3) CARBAMATE CARBONYL CARBON
- **COOO** : (MMFF number 3) C ARBONIC ACID OR ESTER CARBONYL CARBON
- **C=OS** : (MMFF number 3) THIOESTER CARBONYL CARBON, DOUBLE BONDED TO O
- **C=S** : (MMFF number 3) THIOESTER CARBON, DOUBLY BONDED TO S
- **C=SN** : (MMFF number 3) THIOAMIDE, CARBON, DOUBLY BONDED TO S
- **CSO2** : (MMFF number 3) CARBON IN >C=SO2
- **CS=O** : (MMFF number 3) CARBON IN >C=S=O (SULFINYL GROUP)
- **CSS** : (MMFF number 3) THIOCARBOXYLIC ACID OR ESTER CARBONYL CARBON
- **C=P** : (MMFF number 3) CARBON DOUBLE BONDED TO PHOSPHOROUS
- **CSP** : (MMFF number 4) ACETYLENIC CARBON
- **=C=** : (MMFF number 4) ALLENIC CARBON
- **CR4R** : (MMFF number 20) CARBON IN 4-MEMBERED RINGS
- **CR3R** : (MMFF number 22) CARBON IN A 3-MEMBERED RING
- **CE4R** : (MMFF number 30) OLEFINIC CARBON IN 4-MEMBERED RINGS
- **CB** : (MMFF number 37) CARBON AS IN BENZENE, PYRROLE
- **CO2M** : (MMFF number 41) CARBOXYLATE ANION CARBON
- **CS2M** : (MMFF number 41) CARBON IN THIOCARBOXYLATE ANION
- **CGD+** : (MMFF number 57) GUANIDINIUM CARBON
- **CNN+** : (MMFF number 57) C IN +N=C-N RESONANCE STRUCTURES
- **C%** : (MMFF number 60) ISONITRILE CARBON
- **C5A** : (MMFF number 63) ALPHA CARBON IN 5-MEMBERED HETEROAROMATIC RING
- **C5B** : (MMFF number 64) BETA CARBON IN 5-MEMBERED HETEROAROMATIC RING
- **C5** : (MMFF number 78) GENERAL CARBON IN 5-MEMBERED HETEROAROMATIC RING
- **CIM+** : (MMFF number 80) C IN N-C-N IN IMIDAZOLIUM ION

Chlorine
- **CL** : (MMFF number 12) CHLORINE
- **CLO4** : (MMFF number 77) CHLORINE IN PERCHLORATE ANION, CLO4(-)
- **CL-** : (MMFF number 90) CHLORIDE ANION

Copper
- **CU+1** : (MMFF number 97) MONOPOSITIVE COPPER
- **CU+2** : (MMFF number 98) DIPOSITIVE COPPER

Fluorine
- **F** : (MMFF number 11) FLUORINE
- **F-** : (MMFF number 89) FLUORIDE ANION

Hydrogen

- **HC** : (MMFF number 5) H ATTACHED TO C
- **HSI** : (MMFF number 5) H ATTACHED TO SI
- **HOR** : (MMFF number 21) HYDROGEN IN ALCOHOLS
- **HO** : (MMFF number 21) GENERAL H ON OXYGEN
- **HOM** : (MMFF number 21) HYDROGEN IN HYDROXIDE ANION
- **HNR** : (MMFF number 23) H-N(SP3)
- **H3N** : (MMFF number 23) H-N(SP3), AMMONIA
- **HPYL** : (MMFF number 23) H-N IN PYRROLE
- **HNOX** : (MMFF number 23) H-N IN IN A N-OXIDE
- **HNM** : (MMFF number 23) H ON DICOORD, NEGATIVELY CHARGED NITROGEN
- **HN** : (MMFF number 23) GENERAL H ON NITROGEN
- **HOCO** : (MMFF number 24) H-O IN CARBOXYLIC ACIDS
- **HOP** : (MMFF number 24) HYDROGEN ON OXYGEN ATTACHED TO PHOSPHOROUS
- **HN=N** : (MMFF number 27) AZO HYDROGEN
- **HN=C** : (MMFF number 27) IMINE HYDROGEN
- **HNCO** : (MMFF number 28) AMIDE HYDROGEN
- **HNCS** : (MMFF number 28) THIOAMIDE HYDROGEN
- **HNCC** : (MMFF number 28) H-N IN ENAMINES
- **HNCN** : (MMFF number 28) H-N IN H-N-C=N
- **HNNC** : (MMFF number 28) H-N IN H-N-N=C
- **HNNN** : (MMFF number 28) H-N IN H-N-N=N
- **HNSO** : (MMFF number 28) H-N IN SULFONAMIDE
- **HNPO** : (MMFF number 28) H-N IN PHOSPHONAMIDE
- **HNC%** : (MMFF number 28) HYDROGEN ON N ATTACHED TO TRIPLY BONDED CARBON
- **HSP2** : (MMFF number 28) GENERAL H ON SP2 NITROGEN
- **HOCC** : (MMFF number 29) H-O IN ENOLS AND PHENOLS
- **HOCN** : (MMFF number 29) H-O IN HO-C=N
- **HOH** : (MMFF number 31) HYDROGEN IN H2O
- **HOS** : (MMFF number 33) H ON OXYGEN ATTACHED TO SULFUR
- **HNR+** : (MMFF number 36) H ON QUATERNARY NITROGEN
- **HIM+** : (MMFF number 36) H ON IMIDAZOLIUM-TYPE NITROGEN
- **HPD+** : (MMFF number 36) H ON PROTONATED PYRIDINE NITROGEN
- **HNN+** : (MMFF number 36) H ON AMIDINIUM-TYPE NITROGEN
- **HNC+** : (MMFF number 36) H ON PROTONATED IMINE NITROGEN
- **HGD+** : (MMFF number 36) H ON GUANIDINIUM-TYPE NITROGEN
- **HN5+** : (MMFF number 36) H ON N5+, N5A+ OR N5B+
- **HO+** : (MMFF number 50) HYDROGEN ON O+ OXYGEN
- **HO=+** : (MMFF number 52) HYDROGEN ON OXENIUM OXYGEN
- **HS** : (MMFF number 71) H ATTACHED TO DIVALENT, DICOORDINATE S
- **HS=N** : (MMFF number 71) H ATTACHED TO TETRAVALENT, TRICOODR S DBL BONDED
- **HP** : (MMFF number 71) H ATTACHED TO TRI- OR TETRACOORDINATE PHOSPHORUS

Iodine

- **I** : (MMFF number 14) IODINE

Iron

- **FE+2** : (MMFF number 87) IRON +2 CATION
- **FE+3** : (MMFF number 88) IROM +3 CATION

Lithium

- **LI+** : (MMFF number 92) LITHIUM CATION

Magnesium

- **MG+2** : (MMFF number 99) DIPOSITIVE MAGNESIUM CATION

Nitrogen

- **NR** : (MMFF number 8) NITROGEN IN ALIPHATIC AMINES
- **N=C** : (MMFF number 9) NITROGEN IN IMINES
- **N=N** : (MMFF number 9) NITROGEN IN AZO COMPOUNDS
- **NC=O** : (MMFF number 10) NITROGEN IN AMIDES
- **NC=S** : (MMFF number 10) NITROGEN IN N-C=S, THIOAMIDE
- **NN=C** : (MMFF number 10) NITROGEN IN N-N=C
- **NN=N** : (MMFF number 10) NITROGEN IN N-N=N
- **NR+** : (MMFF number 34) QUATERNARY NITROGEN, SP3, POSITIVELY CHARGED
- **NPYD** : (MMFF number 38) NITROGEN, AS IN PYRIDINE
- **NPYL** : (MMFF number 39) NITROGEN, AS IN PYRROLE
- **NC=C** : (MMFF number 40) NITROGEN ON N-C=C
- **NC=N** : (MMFF number 40) NITROGEN IN N-C=N
- **NC=P** : (MMFF number 40) NITROGEN IN N-C=P
- **NC%C** : (MMFF number 40) NITROGEN ATTACHED TO C-C TRIPLE BOND
- **NSP** : (MMFF number 42) NITROGEN, TRIPLE BONDED
- **NSO2** : (MMFF number 43) NITROGEN IN SULFONAMIDES
- **NSO3** : (MMFF number 43) NITROGEN IN SULFONAMIDES, THREE O'S ON S
- **NPO2** : (MMFF number 43) NITROGEN IN PHOSPHONAMIDES
- **NPO3** : (MMFF number 43) NITROGEN IN PHOSPHONAMIDES, THREE O'S ON P
- **NC%N** : (MMFF number 43) NITROGEN ATTACHED TO CYANO GROUP
- **NO2** : (MMFF number 45) NITRO GROUP NITROGEN
- **NO3** : (MMFF number 45) NITRATE GROUP NITROGEN
- **N=O** : (MMFF number 46) NITROSO NITROGEN
- **NAZT** : (MMFF number 47) TERMINAL NITROGEN IN AZIDO OR DIAZO GROUP
- **NSO** : (MMFF number 48) DIVALENT NITROGEN REPLACING MONOVALENT O IN SO2 G
- **=N=** : (MMFF number 53) NITROGEN IN C=N=N OR -N=N=N
- **N+=C** : (MMFF number 54) POSITIVELY CHARGED IMINIUM NITROGEN
- **N+=N** : (MMFF number 54) POSITIVELY CHARGED NITROGEN DOUBLE-BONDED TO N
- **NCN+** : (MMFF number 55) N IN +N=C-N RESONANCE STRUCTURES - FORMAL CHARGE=
- **NGD+** : (MMFF number 56) GUANIDINIUM-TYPE NITROGEN - FORMAL CHARGE=1/3
- **NPD+** : (MMFF number 58) PYRIDINIUM-TYPE NITROGEN - FORMAL CHARGE=1
- **NR%** : (MMFF number 61) ISONITRILE NITROGEN [FC = 0] OR DIAZO NITROGEN [F
- **NM** : (MMFF number 62) DEPROTONATED SULFONAMIDE N-; FORMAL CHARGE=-1
- **N5A** : (MMFF number 65) ALPHA AROM HETEROCYCLIC 5-RING NITROGEN
- **N5B** : (MMFF number 66) BETA AROM HETEROCYCLIC 5-RING NITROGEN
- **N2OX** : (MMFF number 67) SP2-HYDRIDIZED N-OXIDE NITROGEN
- **N3OX** : (MMFF number 68) SP3-HYDRIDIZED N-OXIDE NITROGEN
- **NPOX** : (MMFF number 69) PYRIDINE N-OXIDE NITROGEN
- **N5M** : (MMFF number 76) NEGATIVELY CHARGED N IN, E.G, TRI- OR TETRAZOLE A
- **N5** : (MMFF number 79) GENERAL NITROGEN IN 5-MEMBERED HETEROCYCLIC RING
- **NIM+** : (MMFF number 81) IMIDAZOLIUM-TYPE NITROGEN - FORMAL

CHARGE=1/2

- **N5A+** : (MMFF number 81) POSITIVE N5A NITROGEN - FORMAL CHARGE=1
- **N5B+** : (MMFF number 81) POSITIVE N5B NITROGEN - FORMAL CHARGE=1
- **N5+** : (MMFF number 81) POSITIVE N5 NITROGEN - FORMAL CHARGE=1
- **N5AX** : (MMFF number 82) N-OXIDE NITROGEN IN 5-RING ALPHA POSITION
- **N5BX** : (MMFF number 82) N-OXIDE NITROGEN IN 5-RING BETA POSITION
- **N5OX** : (MMFF number 82) N-OXIDE NITROGEN IN GENERAL 5-RING POSITION

Oxygen

- **OR** : (MMFF number 6) ALCOHOL OR ETHER OXYGEN
- **OC=O** : (MMFF number 6) ESTER OR CARBOXYLIC ACID -O-
- **OC=C** : (MMFF number 6) ENOLIC OR PHENOLIC OXYGEN
- **OC=N** : (MMFF number 6) DIVALENT OXYGEN
- **OC=S** : (MMFF number 6) THIOESTER OR THIOACID -O-
- **ONO2** : (MMFF number 6) DIVALENT NITRATE "ETHER" OXYGEN
- **ON=O** : (MMFF number 6) DIVALENT NITRITE "ETHER" OXYGEN
- **OSO3** : (MMFF number 6) DIVALENT OXYGEN ATTACHED TO SULFUR
- **OSO2** : (MMFF number 6) DIVALENT OXYGEN ATTACHED TO SULFUR
- **OSO** : (MMFF number 6) DIVALENT OXYGEN ATTACHED TO SULFUR
- **OS=O** : (MMFF number 6) DIVALENT OXYGEN ATTACHED TO SULFOXIDE SULFUR
- **-OS** : (MMFF number 6) GENERAL DIVALENT OX ATTACHED TO S
- **OPO3** : (MMFF number 6) DIVALENT OXYGEN ATTACHED TO PHOSPHOROUS
- **OPO2** : (MMFF number 6) DIVALENT OXYGEN ATTACHED TO PHOSPHOROUS
- **OPO** : (MMFF number 6) DIVALENT OXYGEN ATTACHED TO PHOSPHOROUS
- **-OP** : (MMFF number 6) DIVALENT OXYGEN ATTACHED TO PHOSPHOROUS
- **-O-** : (MMFF number 6) GENERAL DIVALENT O
- **O=C** : (MMFF number 7) GENERAL C=O
- **O=CN** : (MMFF number 7) CARBONYL OXYGEN, AMIDES
- **O=CR** : (MMFF number 7) CARBONYL OXYGEN, ALDEHYDES AND KETONES
- **O=CO** : (MMFF number 7) CARBONYL OXYGEN, CARBOXYLIC ACIDS AND ESTERS
- **O=N** : (MMFF number 7) NITROSO OXYGEN
- **O=S** : (MMFF number 7) O=S IN SULFOXIDES
- **O=S=** : (MMFF number 7) O=S ON SULFUR DOUBLY BONDED TO, E.G., CARBON
- **O2CM** : (MMFF number 32) OXYGEN IN CARBOXYLATE ANION
- **OXN** : (MMFF number 32) N-OXIDE OXYGEN
- **O2N** : (MMFF number 32) NITRO OXYGEN
- **O2NO** : (MMFF number 32) NITRO-GROUP OXYGEN IN NITRATE
- **O3N** : (MMFF number 32) NITRATE ANION OXYGEN
- **O-S** : (MMFF number 32) SINGLE TERMINAL OXYGEN ON TETRACOORD SULFUR
- **O2S** : (MMFF number 32) TERMINAL O-S IN SULFONES AND SULFONAMIDES
- **O3S** : (MMFF number 32) TERMINAL O IN SULFONATES
- **O4S** : (MMFF number 32) TERMINAL O IN SO4(-3)
- **OSMS** : (MMFF number 32) TERM O IN THIOSULFINATE ANION - FORMAL CHARGE=-0.
- **OP** : (MMFF number 32) TERMINAL O IN PHOSPHOXIDES
- **O2P** : (MMFF number 32) TERMINAL O IN PHOSPHINATES
- **O3P** : (MMFF number 32) TERMINAL OXYGEN IN PHOSPHONATES
- **O4P** : (MMFF number 32) TERMINAL OXYGEN IN PHOSPHATES AND PHOSPHODIESTERS
- **O4CL** : (MMFF number 32) OXYGEN IN CLO4(-) ANION - FORMAL CHARGE=-

0.25

- **OM** : (MMFF number 35) ALKOXIDE OXYGEN, NEGATIVELY CHARGED
- **OM2** : (MMFF number 35) OXIDE OXYGEN ON SP2 CARBON, NEGATIVELY CHARGED
- **O+** : (MMFF number 49) POSITIVELY CHARGED OXONIUM (TRICOORDINATE) OXYGEN
- **O=+** : (MMFF number 51) POSITIVELY CHARGED OXENIUM (DICOORDINATE) OXYGEN
- **OFUR** : (MMFF number 59) AROMATIC OXYGEN AS IN FURAN
- **OH2** : (MMFF number 70) OXYGEN ON WATER

Phosphorus

- **PO4** : (MMFF number 25) PHOSPHOROUS IN PHOSPHATES AND PHOSPHODIESTERS
- **PO3** : (MMFF number 25) TETRACOORDINATE P WITH THREE ATTACHED OXYGENS
- **PO2** : (MMFF number 25) TETRACOORDINATE P WITH TWO ATTACHED OXYGENS
- **PO** : (MMFF number 25) TETRACOORDINATE P WITH ONE ATTACHED OXYGEN
- **PTET** : (MMFF number 25) GENERAL TETRACOORDINATE PHOSPHORUS
- **P** : (MMFF number 26) TRICOORDINATE P, AS IN PHOSPHINES
- **-P=C** : (MMFF number 75) PHOSPHOROUS DOUBLY BONDED TO CARBON

Potassium

- **K+** : (MMFF number 94) POTASSIUM CATION

Silicon

- **SI** : (MMFF number 19) SILICON

Sodium

- **NA+** : (MMFF number 93) SODIUM CATION

Sulfur

- **S** : (MMFF number 15) SULFUR IN THIOETHERS AND MERCAPTANS
- **S=C** : (MMFF number 16) TERMINAL SULFUR DOUBLY BONDED TO CARBON
- **S=O** : (MMFF number 17) SULFUR IN SULFOXIDES
- **>S=N** : (MMFF number 17) SULFUR, TRICOORD, DOUBLY BONDED TO N
- **SO2** : (MMFF number 18) SULFUR IN SULFONES
- **SO2N** : (MMFF number 18) SULFUR IN SULFONAMIDES
- **SO3** : (MMFF number 18) SULFONATE SULFUR
- **SO4** : (MMFF number 18) SULFATE SULFUR
- **=SO2** : (MMFF number 18) SULFONE SULPHER DOUBLY BONDED TO CARBON
- **SNO** : (MMFF number 18) SULFUR IN NITROGEN ANALOG OF A SULFONE
- **STHI** : (MMFF number 44) SULFUR AS IN THIOPHENE
- **S-P** : (MMFF number 72) TERMINAL SULFUR BONDED TO PHOSPHORUS
- **S2CM** : (MMFF number 72) TERMINAL SULFUR IN THIOCARBOXYLATE ANION
- **SM** : (MMFF number 72) TERMINAL SULFUR - FORMAL CHARGE=-1
- **SSMO** : (MMFF number 72) TERMINAL SULFUR IN THIOSULFINATE GROUP
- **SO2M** : (MMFF number 73) SULFUR IN NEGATIVELY CHARGED SULFINATE GROUP
- **SSOM** : (MMFF number 73) TRICOORD SULFUR IN THIOSULFINATE GROUP
- **=S=O** : (MMFF number 74) SULFINYL SULFUR, EG. IN C=S=O

Zinc

- **ZINC** : (MMFF number 95) DIPOSITIVE ZINC
- **ZN+2** : (MMFF number 95) DIPOSITIVE ZINC

Coulombic interactions

This force field uses point charges and has been set up to assign the point charges using the bond increment method.

Return to the Towhee Capabilities web page

Send comments to: Marcus G. Martin

Last updated: December 16, 2006

MCCCS Towhee (Richards *et al.* 1995)

Overview

This section covers the Richards *et al.* 1995 (Richar1995) force field as it is implemented into the towhee_ff_Richar1995 file in the ForceFields directory. This force field was designed to reproduce the adsorption behavior of N_2 and O_2 in Zeolite Li-X. Note that this is a Lennard-Jones force field that uses the 'explicit' mixing rules, and therefore is not easily combined with other forcefields. Any discrepencies (especially typos) from the published values are the sole responsibility of Marcus G. Martin, and I welcome feedback on how this implementation compares with other programs.

Reference for Richards *et al.* 1995

This forcefield is described in a single paper.

- Richards *et al.* 1995

Typos and comments for Richards *et al.* 1995

- I have been unable to reproduce the results shown in Figure~4 and have observed substantially higher adsorption for both N_2 and O_2 using this implementation. Either the Lithium ion placement is crucial for adsorption in this zeolite, or there is an error in the results presented in their paper. Either way, the user is advised to use caution with this forcefield and if anyone is able to reproduce those results please let the developers know.

Richards *et al.* 1995 in Towhee

The official force field name for Richards *et al.* 1995 in Towhee is 'Richar1995'. Here is the list of atom names for use in the towhee_input file, along with a brief description. Please note that the capitalization and spacing pattern is important and must be followed exactly as listed here. Note that both N_2 and O_2 are modelled with three interaction sites in this potential.

- **'O'** : Oxygen in O_2 that is bonded to the midpoint site 'mp'
- **'N'** : Nitrogen in N_2 that is bonded to the midpoint site 'mp'
- **'Si'** : Silicon in zeolite Li-X
- **'Al'** : Aluminum in zeolite Li-X
- **'Oz'** : Oxygen in zeolite Li-X
- **'Li'** : Lithium in zeolite Li-X
- **'mp'** : the midpoint charge site of either O_2 or N_2

Coulombic interactions

This force field uses point charges and has been set up to assign the point charges using the bond increment method.

Return to the Towhee Capabilities web page

Send comments to: Marcus G. Martin
Last updated: October 1, 2008

MCCCS Towhee (MCY1976)

Overview

This section covers the MCY1976 force field that has two similar four-site water models. This forcefield is implemented into the towhee_ff_MCY1976 file in the ForceFields directory. All of the Towhee atom types for this force field are listed, along with a short description of their meanings. MCY1976 has two different rigid water models with 'Double Exponential' terms on the hydrogens and oxygens and charge sites on the hydrogens and along the bisector between the hydrogens. These are branched, rigid molecules (when you consider the charged interaction site) and it is recommended to generate initial structures from a template and also to use the rotational-bias moves instead of the configurational-bias moves for molecule transfers between simulation boxes. Any discrepencies (especially typos) from the published force field values are the sole responsibility of Marcus Martin, and we welcome feedback on how this implementation compares with other programs.

References for MCY1976

MCY1976 was originally published in the following paper

- Matsuoka *et al.* 1976

MCY1976 in Towhee

The official force field name for this potential in Towhee is 'MCY1976'. This section lists all of the atom names for use in the towhee_input file, along with a brief description of the atom names. Please note that the capitalization and spacing pattern is important and must be followed exactly as listed here.

- **'H:CI'** : hydrogen in water using the model they fit to the CI data
- **'O:CI'** : oxygen in water using the model they fit to the CI data
- **'M:CI'** : charge site along the bisector between the hydrogens using the model they fit to the CI data
- **'H:inter'** : hydrogen in water using the model they fit to the inter data
- **'O:inter'** : oxygen in water using the model they fit to the inter data
- **'M:inter'** : charge site along the bisector between the hydrogens using the model they fit to the inter data

Coulombic interactions

This forcefield uses atom-centered point charges to represent the electrostatic interactions. The 'bond increment' method is implemented for this potential and assigns charges according to Table VIII of their original paper.

Return to the Towhee Capabilities web page

Send comments to: Marcus G. Martin

Last updated: January 05, 2007

MCCCS Towhee (Ackland et al. 2004)

Overview

This section covers the Ackland *et al.* 2004 iron and phosphorus parameters as they are implemented into the towhee_ff_Ackl2004 file in the ForceFields directory. All of the Towhee atom types for this force field are listed, along with a short description of their meanings. This force field was intended for solid and liquid simulation of iron with small amounts of phosphorus impurities, uses the 'Embedded Atom Method' potential, and is not easily combined with other potentials due to the explicit mixing rules used with that potential. I would like to acknowledge Graeme Ackland who provides an electronic copy of these parameters on his web page. Any discrepencies (especially typos) from the published values are the sole responsibility of Marcus G. Martin, and I welcome feedback on how this implementation compares with other programs.

References for Ackl2004

- Ackland *et al.* 2004

Ackland *et al.* 2004 in Towhee

The official force field name for these parameters is 'Ackl2004'. Here I list the atom names for use in the towhee_input file, along with a brief description (although the meaning of the atom names should be obvious). There were two parameterizations presented in that paper and they are both implemented here. Please note that the capitalization and spacing pattern is important and must be followed exactly as listed here.

- **'Fe'** : iron. Uses a 5.3 Angstrom cutoff.
- **'P'** : phosphorus. Uses a 5.3 Angstrom cutoff.

Coulombic interactions

There are no coulombic interactions for this potential.

Return to the Towhee Capabilities web page

Send comments to: Marcus G. Martin
Last updated: September 07, 2005

MCCCS Towhee (Hoyt et al. Cu and Pb)

Overview

This section covers the Hoyt *et al.* copper and lead parameters as they are implemented into the towhee_ff_Hoyt2003 file in the ForceFields directory. All of the Towhee atom types for this force field are listed, along with a short description of their meanings. This force field was intended for lead and copper. Any discrepencies (especially typos) from the published values are the sole responsibility of Marcus G. Martin, and I welcome feedback on how this implementation compares with other programs.

References for Hoyt *et al.* Pb and Cu

- Hoyt *et al.* 2003
- Foiles *et al.* 1986 (original source for Cu)
- Lim *et al.* 1992 (original source for Pb)

Hoyt *et al.* in Towhee

The official force field name for these parameters is 'Hoyt2003'. Here I list the atom names for use in the towhee_input file, along with a brief description (although the meaning of the atom names should be obvious). Please note that the capitalization and spacing pattern is important and must be followed exactly as listed here.

- **'Cu'** : copper
- **'Pb'** : lead

Coulombic interactions

There are generally no coulombic interactions for this potential.

Improper torsions

This force field does not have improper torsions.

Proteins

This force field does not have parameters for proteins.

Return to the Towhee Capabilities web page

Send comments to: Marcus G. Martin
Last updated: September 07, 2005

MCCCS Towhee (Mendelev et al. 2003)

Overview

This section covers the Mendelev *et al.* 2003 iron parameters as they are implemented into the towhee_ff_Mend2003 file in the ForceFields directory. All of the Towhee atom types for this force field are listed, along with a short description of their meanings. This force field was intended for solid and liquid simulation of iron and uses the 'Embedded Atom Method' potential and is not easily combined with other potentials due to the explicit mixing rules used with that potential. I would like to acknowledge Mikhail Mendelev for kindly providing me with an electronic copy of these parameters. Any discrepencies (especially typos) from the published values are the sole responsibility of Marcus G. Martin, and I welcome feedback on how this implementation compares with other programs.

References for Mend2003

- Mendelev *et al.* 2003

Typos and clarifications for Mend2003

- a_7^{phi} in Table A 1 is listed incorrectly and should be negative instead of positive, as noted on Graeme Ackland's metals page.

Mendelev *et al.* 2003 in Towhee

The official force field name for these parameters is 'Mend2003'. Here I list the atom names for use in the towhee_input file, along with a brief description (although the meaning of the atom names should be obvious). There were two parameterizations presented in that paper and they are both implemented here. Please note that the capitalization and spacing pattern is important and must be followed exactly as listed here.

- **'Fe-p2'** : iron potential 2. Uses a 5.2 Angstrom cutoff.
- **'Fe-p4'** : iron potential 4. Uses a 6.0 Angstrom cutoff.

Coulombic interactions

There are no coulombic interactions for this potential.

Return to the Towhee Capabilities web page

Send comments to: Marcus G. Martin
Last updated: September 07, 2005

MCCCS Towhee (Catlow/Faux)

Overview

This section covers the Catlow and Faux force field as it is implemented into the towhee_ff_CatlowFaux file in the ForceFields directory. All of the Towhee atom types for the Catlow and Faux force field are listed, along with a short description of their meanings. Note that the Catlow and Faux force fields are a hybrid of Lennard-Jones (12-6) and an exponential replusion and cannot be combined with any other force fields. I would like to acknowledge David A. Faux for providing very useful guidance about implementing Catlow and Faux force field. Any discrepencies (especially typos) from the published force field values are the sole responsibility of Marcus G. Martin, and I welcome feedback on how this implementation compares with other programs.

References for Catlow and Faux

This force field comes from a variety of sources. Several authors have used zeolite parameters that are similar (but often slightly different) to those published in

- Jackson and Catlow 1988

The implementation in Towhee uses the zeolite (OZ, Al, Si, Na) parameters as modified in

- Faux *et al.* 1997

with the exception that there is a typo in Table 1 of that paper which listed harmonic terms for the Al-OZ-Si and Si-OZ-Al angles. That should instead be harmonic parameters for the OZ-Al-OZ and OZ-Si-OZ angles. Note that these harmonic terms are in addition to the normal van der Waals and coulombic interaction between these atoms. There is no additional harmonic term for either the Al-OZ-Si or Si-OZ-Al (personal communication between M.G. Martin and D.A. Faux February 13, 2001).
The hydrocarbon (Cc and Hc) nonbonded parameters, and the cross terms with OZ, Al, Si (note the cross terms are set to zero for Cc-Si, Cc-Al, Hc-Si, and Hc-Al) are from

- Raj *et al.* 1999

However, we wanted a fully flexible alkane model so we used the bonded parameters from

- Catlow *et al.* 1991

instead of the semi-rigid model from Raj *et al.* 1999.
I also concocted some cross terms for Na-Cc and Na-Hc by combining Raj *et al.* 1999 with sodium parameters from

- Aqvist 1990

using geometric mixing rules.

Catlow/Faux in Towhee

The official force field name for these parameters is 'CatlowFaux' (starting with version 3.2.9)
Here I list all of the Catlow and Faux atom names for use in the towhee_input file, along with a brief description of what atoms we are talking about. Please note that the capitalization and spacing pattern is important and must be followed exactly as listed here.

- **'Cc'** : carbon in an alkane
- **'Hc'** : hydrogen bonded to carbon in an alkane
- **'Al'** : aluminum in a zeolite
- **'Na'** : sodium ion
- **'OZ'** : oxygen in a zeolite
- **'Si'** : silicon in a zeolite

Coulombic Interactions

This force field uses point charges on atomic centers. Here we list suggested charges for each of the atom types in this force field.
Catlow charges are only listed for n-octane in Raj *et al.* 1999. It is not obvious how to interpolate those charges for other alkanes, but it would be consistent with the reasoning in that paper to set all of the Hydrogens to 0.100 and then evenly divide the counter charge across all of the carbons.

- **'Cc'** : In n-octane this is set to -0.225. One possible way to do this for general alkanes would be a charge equal to the number of hydrogens times the charge on each hydrogen divided by the total number of carbons.
- **'Hc'** : In n-octane this is set to 0.100. One possible way to do this for general alkanes would be to always set the Hc charge to 0.100 and adjust the counter charge on the carbons.

Faux charges from Faux *et al.* 1997

- **'Al'** : 2.775
- **'Na'** : 1.0
- **'OZ'** : -1.86875
- **'Si'** : 3.7

Improper torsions

There are no improper torsions in the Catlow or Faux force fields.

Proteins

The Catlow and Faux force fields are designed for zeolites and are not suited for proteins.

Return to the Towhee Capabilities web page

Send comments to: Marcus G. Martin
Last updated: September 07, 2005

MCCCS Towhee (Kramer-Farragher-van Beest-van Santen)

Overview

This section covers the Kramer-Farragher-van Beest-van Santen (KFvBvS) force field as it is implemented into the towhee_ff_KFvBvS file in the ForceFields directory. All of the Towhee atom types for the KFvBvS force field are listed, along with a short description of their meanings. The KFvBvS force field uses the exponential-6 potential type and therefore must also use explicit combining rules. It therefore cannot be combined with other force fields. Any discrepencies (especially typos) from the published force field values are the sole responsibility of Marcus G. Martin, and I welcome feedback on how this implementation compares with other programs.

References for KFvBvS

This force field comes from the work of Kramer, Farragher, van Beest, and van Santen. I created the name for this force field from the initials of the last names of the authors. The parameters were taken from the mixed SCF empirical force field listed in

- van Beest *et al.* 1990
- Kramer *et al.* 1991

KFvBvS in Towhee

The official force field name for these parameters is either 'KFvBvS'. Here I list all of the KFvBvS atom names for use in the towhee_input file, along with a brief description of what atoms we are talking about. This force field is intended for use with silacious materials (zeolites, silica, etc.). For some reason they did not include repulsive potentials between many of the atoms and Cl. Thus, Cl can only be used in combination with other negative ions, or with Na. Please note that the capitalization and spacing pattern is important and must be followed exactly as listed here.

- **'Al'** : aluminum
- **'Cl'** : chlorine
- **'Na'** : sodium
- **'O'** : oxygen
- **'P'** : phosphorous
- **'Si'** : silicon

Coulombic interactions

Atom centered point charges are used to represent the electrostatic interactions. Below is a list of charges taken from the mixed SCF empirical force field in Kramer *et al.* 1991

- **'Al'** : 1.4
- **'Cl'** : -1.0
- **'Na'** : 1.0
- **'O'** : -1.2
- **'P'** : 3.4
- **'Si'** : 2.4

Improper torsions

There are no improper torsions in this force field.

Proteins

The KFvBvS force field is designed for zeolites and is not suited for proteins.

Return to the Towhee Capabilities web page

MCCCS Towhee (MM2)

Overview

This section covers the MM2 force field as it is implemented into the towhee_ff_MM2 file in the ForceFields directory. This force field was originally designed for to reproduce hydrocarbon crystal data. Note that this is an Exponential-6 force field that uses explicit mixing rules so it is not easily combined with other force fields. Any discrepencies (especially typos) from the published values are the sole responsibility of Marcus G. Martin, and I welcome feedback on how this implementation compares with other programs.

References for MM2

MM2 is described in the following paper.

- Allinger 1977

MM2 in Towhee

The official force field name for MM2 in Towhee is 'MM2'. Here I list all of the atom names for use in the towhee_input file, along with a brief description. I created these atom names in order to work with the molecule assembler. Please note that the capitalization and spacing pattern is important and must be followed exactly as listed here.

- **'Ccc**(sp3)'** : Carbon single bonded to 2 or more other carbons.
- **'Cchhh(sp3)'** : Carbon single bonded to 1 carbon and 3 hydrogens.
- **'H'** : hydrogen.

Coulombic interactions

This force field does not use coulombic interactions.

Return to the Towhee Capabilities web page

Send comments to: Marcus G. Martin
Last updated: September 07, 2005

MCCCS Towhee (Gordon)

Overview

This section covers a force field parameterized to work with the Gordon n-6 potential. Right now this is mostly a placeholder, but efforts are currently underway to determine parameters for several functional groups. Check back later and see if any progress has occurred.

References for Gordon

Nothing yet to report.

Gordon in Towhee

The official force field name for Gordon in Towhee is 'Gordon'. Here I list all of the atom names for use in the towhee_input file, along with a brief description. Please note that the capitalization and spacing pattern is important and must be followed exactly as listed here.

- **'CH4_3'** : methane taken from the TraPPE-UA potential

Coulombic interactions

Nothing yet to report.

Return to the Towhee Capabilities web page

Send comments to: Marcus G. Martin
Last updated: September 07, 2005

MCCCS Towhee (Cui *et al.* 1998)

Overview

This section covers the Cui *et al.* 1998 (Cui1998) force field as it is implemented into the towhee_ff_Cui1998 file in the ForceFields directory. This force field contains two different parameterizations designed for vapor-liquid properties of perfluorinated linear alkanes. Note that this is a Lennard-Jones force field that uses the 'Lorentz-Berthelot' mixing rules. Any discrepencies (especially typos) from the published values are the sole responsibility of Marcus G. Martin, and I welcome feedback on how this implementation compares with other programs.

References for Cui *et al.* 1998

This forcefield is described in a single paper.

- Cui *et al.* 1998

Typos and comments for Cui *et al.* 1998

- The T version of this force field was parameterized for use with an 11.5 Angstrom cutoff and analytical tail corrections.
- The M version of this force field was parameterized for use with a 14 Angstrom cutoff without any shifting or tail corrections.

Cui *et al.* 1998 in Towhee

The official force field name for Cui *et al.* 1998 in Towhee is 'Cui1998'. Here is the list of atom names for use in the towhee_input file, along with a brief description. These names correspond to those used in their paper and I follow their convention that the two different parameterizations are labeled T and M for the version of the Tennessee group, and the version of the Minnesota group, respectively. Please note that the capitalization and spacing pattern is important and must be followed exactly as listed here.

- **'CF3(M)'** : a united-atom of carbon and three fluorines that is centered on the carbon: Minnesota version.
- **'CF2(M)'** : a united-atom of carbon and two fluorines that is centered on the carbon: Minnesota version.
- **'CF3(T)'** : a united-atom of carbon and three fluorines that is centered on the carbon: Tennessee version.
- **'CF2(T)'** : a united-atom of carbon and two fluorines that is centered on the carbon: Tennessee version.

Coulombic interactions

They did not use any partial charges with this forcefield.

Return to the Towhee Capabilities web page

Send comments to: Marcus G. Martin
Last updated: May 02, 2006

MCCCS Towhee (Cui and Elliott 2002)

Overview

This section covers Cui and Elliott 2002 parameters as they are implemented into the towhee_ff_Cui2002 file in the ForceFields directory. This force field contains parameters for linear alkanes and uses the 'Hard 2580 Multistep' force field. It is designed for use with the Lorentz-Berthelot mixing rule. I would like to acknowledge J. Richard Elliott for providing useful guidance about implementing this force field. Any discrepencies (especially types) from the published force field values are the sole responsibility of Marcus G. Martin, and I welcome feedback on how this implementation compares with other programs.

References for Cui and Elliott 2002

- Cui and Elliott 2002

Cui and Elliott 2002 in Towhee

The official force field name for Cui and Elliott 2002 in Towhee is 'Cui2002'. Here is a list of all atom names currently in use for the towhee_input file, along with a brief description. Please note that the capitalization and spacing pattern is important and must be followed exactly as listed here.

- **'CH3'** : united-atom CH_3 group in an *n*-alkane
- **'CH2'** : united-atom CH_2 group in an *n*-alkane

Coulombic interactions

This force field does not utilize coulombic interactions.

Return to the Towhee Capabilities web page

Send comments to: Marcus G. Martin
Last updated: December 03, 2005

MCCCS Towhee (Hard Sphere)

Overview

This section covers Hard Sphere parameters as they are implemented into the towhee_ff_HardSphere file in the ForceFields directory. It is a bit strong to call this a force field, as all there is to the hard spheres is a series of sigma values, but this explains how to use the builder with the hard sphere potentials, and how to add in any other hard sphere diameters that you might want to use in a simulation.

Hard Spheres in Towhee

The official force field name for Hard Spheres in Towhee is 'HardSphere'. Here I list all of the Hard Sphere atom names currently in use for the towhee_input file, along with a brief description (although the meaning of the atom names should be obvious). If you want to add any new hard sphere diameters into Towhee just take a look in the ffhardsphere.F subroutine. Search for one of the atom types listed below and you will find the complete listing of hard sphere parameters. It should be obivous how to add new atom diameters from the examples. Please note that the capitalization and spacing pattern is important and must be followed exactly as listed here.

- **'hs0.0'** : hard sphere with 0.0 diameter. This is essentially an ideal atom, but it is still excluded from being inside of other atoms. Thus it is a hard point potential.
- **'hs0.5'** : hard sphere with 0.5 diameter.
- **'hs1.0'** : hard sphere with 1.0 diameter.

Coulombic interactions

It is possible to combine the hard sphere potential with point charges. Simply assign the point charges to the atoms as you see fit.

Improper torsions

There are no improper torsions set up specifically for hard spheres.

Proteins

There are no features to build proteins out of hard spheres.

Return to the Towhee Capabilities web page

Send comments to: Marcus G. Martin
Last updated: September 07, 2005

MCCCS Towhee (Alavi *et al.* 2005)

Overview

This section covers the Alavi *et al.* 2005 (Alavi2005) force field as it is implemented into the towhee_ff_Alavi2005 file in the ForceFields directory. All of the Towhee atom types for this force field are listed, along with a short description of their meanings. Note that this a Lennard-Jones (12-6) force field and can only be combined with other Lennard-Jones (12-6) force fields. Any discrepencies (especially typos) from the published values are the sole responsibility of Marcus G. Martin, and I welcome feedback on how this implementation compares with other programs.

References for Alavi2005

This force field was published in a single paper where it was used along with the SPC/E force field for water.

- Alavi *et al.* 2005

Alavi *et al.* 2005 in Towhee

The official force field name for Alavi *et al.* 2005 in Towhee is 'Alavi2005'. Here I list all of the atom names for use in the towhee_input file, along with a brief description. I created these atom names in order to work with the molecule assembler. This force field was designed to work with the SPC/E water model. This force field uses a three site model for H_2 as described below. Please note that the capitalization and spacing pattern is important and must be followed exactly as listed here.

- **'H2-com'** : the center of mass of an H_2 molecule. This atom type has no mass itself and should be bonded to 2 **H2-end** sites.
- **'H2-end'** : hydrogen atom in H_2. This atom type represents the mass point for each hydrogen atom and should be bonded to the **H2-com** site.

Coulombic interactions

Alavi2005 assigns a point charge to the H_2 mass sites and also to the center of mass location. The bond increment method is implemented for this forcefield and can be used to automatically assign appropriate charges. Please see the inpstyle 2 documentation for further information on enabling this option. Here I list the bond increments using the bond names listed above. The value of the bond increment is added to the first atom listed and subtracted from the second atom listed.

- H2-end - H2-com: 0.4932

Send comments to: Marcus G. Martin
Last updated: September 26, 2005

MCCCS Towhee (Amber param96)

Overview

This section covers the Amber param96 force field as it is implemented into the Towhee code. All of the Towhee atom types for the Amber param96 force field are listed, along with a short description of their meanings. For more information about the Amber force field see the Amber web site. Note that Amber param96 is a Lennard-Jones (12-6) force field and can only be combined with other Lennard-Jones (12-6) force fields. I would like to acknowledge Wendy D. Cornell and the late Peter A. Kollman for providing very useful guidance about implementing Amber param96. Any discrepencies (especially typos) from the published Amber param96 force field values are the sole responsibility of Marcus G. Martin, and I welcome feedback on how this implementation compares with other programs.

References for Amber param96

The best literature reference for Amber param96 is

- Cornell *et al.* 1995.

An electronic version of the Amber param96 parameters can be found at the Amber web site.

Amber param96 in Towhee

The official force field name for Amber param96 in Towhee is 'Amber96' and the file name is towhee_ff_Amber96 in the ForceFields directory. Here I list all of the Amber96 atom names for use in the towhee_input file, along with a brief description taken from the Cornell *et al.* paper. I have added some comments where I thought clarification was needed, these are all in [square brackets]. Please note that the capitalization and spacing pattern is important and must be followed exactly as listed here.

Carbon

- **'CT'** : any sp^3 carbon
- **'C'** : any carbonyl sp^2 carbon
- **'CA'** : any aromatic sp^2 carbon and ($C_{epsilon}$ of Arg)
- **'CM'** : any sp^2 carbon, double bonded
- **'CC'** : sp^2 aromatic in 5-membered ring with one substituent + next to nitrogen (C_{gamma} in His)
- **'CV'** : sp^2 aromatic in 5-membered ring next to carbon and lone pair nitrogen (e.g. C_{delta} in His(delta))
- **'CW'** : sp^2 aromatic in 5-membered ring next to carbon and NH (e.g. C_{delta} in His(epsilon) and in Trp)
- **'CR'** : sp^2 aromatic in 5-membered ring next to two nitrogens (C_{gamma} and $C_{epsilon}$ in His)
- **'CB'** : sp^2 aromatic at junction of 5- and 6-membered rings (C_{delta} in Trp) and both junction atoms in Ade and Gua
- **'C*'** : sp^2 aromatic in 5-membered ring next to two carbons (e.g. C_{gamma} in Trp)
- **'CN'** : sp^2 junction between 5- and 6-membered rings and bonded to CH and NH ($C_{epsilon}$ in Trp)
- **'CK'** : sp^2 carbon in 5-membered aromatic between N and N-R (C8 in purines)
- **'CQ'** : sp^2 carbon in 6-membered ring between lone pair nitrogens (e.g. C2 in purines)

Chlorine

- **'Cl-'** : ionic chlorine (-1 charge)

Hydrogen

- **'H'** : H attached to N
- **'HA'** : H attached to aromatic carbon with no electronegative neighbors
- **'HC'** : H attached to aliphatic carbon with no electron-withdrawing substituents
- **'H1'** : H attached to aliphatic carbon with one electron-withdrawing substituent
- **'H2'** : H attached to aliphatic carbon with two electron-withdrawing substituents
- **'H3'** : H attached to aliphatic carbon with three electron-withdrawing substituents
- **'HW'** : H in TIP3P water
- **'HO'** : H in alcohols and acids
- **'HS'** : H attached to sulfur
- **'HP'** : H attached to carbon directly bonded to formally positive atoms (e.g C next to NH_3^+ of lysine)
- **'H4'** : H attached to aromatic carbon with one electronegative neighbor (e.g hydrogen on C5 of Trp, C6 of Thy)
- **'H5'** : H attached to aromatic carbon with two electronegative neighbors (e.g H8 of Ade and Gua and H2 of Ade)

Nitrogen

- **'N'** : sp^2 nitrogen in amides
- **'NA'** : sp^2 nitrogen in aromatic rings with hydrogen attached (e.g. protonated His, Gua, Trp)
- **'NB'** : sp^2 nitrogen in 5-membered ring with lone pair (e.g. N7 in purines)
- **'NC'** : sp^2 nitrogen in 6-membered ring with lone pair (e.g. N3 in purines)
- **'N*'** : sp^2 nitrogen in 5-membered ring with carbon substituent (in purine nucleosides)
- **'N2'** : sp^2 nitrogen of aromatic amines and guanidinium ions
- **'N3'** : sp^3 nitrogen

Oxygen

- **'OW'** : sp^3 oxygen in TIP3P water
- **'OH'** : sp^3 oxygen in alcohols, tyrosine, and protonated carboxylic acids
- **'OS'** : sp^3 oxygen in ethers
- **'O'** : sp^2 oxygen in amides
- **'O2'** : sp^2 oxygen in anionic acids

Phosphorous

- **'P'** : phosphorus in phosphates

Sulfur

- **'S'** : sulfur in methionine and cysteine [sulfide form of cysteine]
- **'SH'** : sulfur in cysteine [thiol form of cysteine]

Coulombic Interactions

Amber 96 uses point charges located at atomic centers to represent the elecrostatic interactions between atoms. There is no simple, general table of charges for atoms in this force field. Instead, they recommend using the RESP method to fit the charges for each molecule of interest. See the Amber web site for more information about this method. See the protein section below for comments on the Amber param96 charges used with the protein builder.

Improper torsions

Improper torsions are not automatically generated by the Towhee code as the rules for determining where they are applied are not always straight-forward. Amber param96 exclusively uses the Stereocenter version of the improper torsions, and they are typically centered on an sp^2 atom in order to enforce planarity with its three neighbors. Only one improper torsion allowed to be centered on any atom. These torsions are listed in the Amber literature as i-j-k-l where the angle is the dihedral between i-k-l and j-k-l, and the bonding pattern is i, j, and l are all bonded to atom k, and are also not bonded to each other. In the towhee_input file this stereo improper torsion is listed only for atom k, and the atom order there is l, i, j. Remember that you can set the improper type to 0 to have the code automatically determine the improper type (so long as inpstyle is 2).

Proteins

All of the 20 basic amino acids (including the 3 forms of hystidine and the thiol and disulfide forms of cysteine) are functional for the Amber96 force field. I have implemented the atom types and charges according to the published Amber param96 values except in the following cases.

- N-terminal leucine: the electronic files for Amber param96 list CD1 with a charge of -0.41060 and atom CD2 with a charge of -0.41040. These atoms are symmetric and indistinguishable in reality so I felt it made no sense to give them different charges just because they have different names. Therefore I put a charge of -0.41050 on both atoms.
- N-terminal phenylalanine: the electronic files for Amber param96 list CE1 with a charge of -0.16020 and atom CE2 with a charge of -0.16030. These atoms are symmetric and indistinguishable in reality so I put a charge of -0.16025 on both atoms.

Here is a complete list of the groups for Amber96.

- 'a0' alanine
- 'c0' cysteine with hydrogen on the sulfur
- 'cs' cysteine in a disulfide bond
- 'cp' cysteine bonded to palmitate (includes the palmitate)
- 'd-' aspartic acid deprotonated
- 'e-' glutamic acid deprotonated
- 'f0' phenylalanine
- 'g0' glycine
- 'h+' histidine both N protonated
- 'hd' histidine neutral with only N_d protonated
- 'he' histidine neutral with only N_e protonated
- 'i0' isoleucine
- 'k+' lysine protonated
- 'l0' leucine
- 'm0' methionine
- 'n0' asparagine
- 'p0' proline
- 'q0' glutamine
- 'r+' arginine protonated
- 's0' serine
- 't0' threonine
- 'v0' valine
- 'w0' tryptophan
- 'y0' tyrosine

Return to the Towhee Capabilities web page

Send comments to: Marcus G. Martin
Last updated: May 12, 2006

MCCCS Towhee (Aqvist ions)

Overview

This section covers the Aqvist ion parameters as they are implemented into the towhee_ff_Aqvist file in the ForceFields directory. All of the Towhee atom types for the Aqvist ion force field are listed, along with a short description of their meanings. This is not really a comphrehensive stand-alone force field, but is instead a collection of parameters for the positive ions in the first two columns of the periodic table. Ion parameters are sometimes hard to come by in the literature so I have included these parameters for use in conjunction with other force fields (as a simulation of only positive ions is not very interesting). Note that these are Lennard-Jones (12-6) parameters and can only be combined with other Lennard-Jones (12-6) force fields. Any discrepencies (especially typos) from the published Aqvist force field values are the sole responsibility of Marcus G. Martin, and I welcome feedback on how this implementation compares with other programs.

References for Aqvist ions

- Aqvist 1990

Aqvist in Towhee

The official force field name for Aqvist ions in Towhee is 'Aqvist'. Here I list all of the Aqvist atom names for use in the towhee_input file, along with a brief description (although the meaning of the atom names should be obvious). Please note that the capitalization and spacing pattern is important and must be followed exactly as listed here.

- **'Ba+2'** : barium ion with a +2 charge
- **'Ca+2'** : calcium ion with a +2 charge
- **'Cs+'** : cesium ion with a +1 charge
- **'K+'** : potasium ion with a +1 charge
- **'Li+'** : lithium ion with a +1 charge
- **'Mg+2'** : magnesium ion with a +2 charge
- **'Na+'** : sodium ion with a +1 charge
- **'Rb+'** : rubidium ion with a +1 charge
- **'Sr+2'** : strontium ion with a +2 charge

Coulombic Interactions

The Aqvist ions utilize point charges. All of the atoms in this force field are ions so the charge assignment is obvious.

Improper torsions

This force field only has parameters for ions so there are no improper torsions.

Proteins

This force field only has parameters for ions, and while you are welcome to use them with another protein force field, there are no amino acids in the Aqvist ions force field.

Return to the main towhee web page

Send comments to: Marcus G. Martin *Last updated:* September 07, 2005

MCCCS Towhee (Charmm22)

Overview

This section covers the Charmm22 force field as it is implemented into the towhee_ff_Charmm22 file in the ForceFields directory. All of the Towhee atom types for the Charmm22 force field are listed, along with a short description of their meanings. For more information about the Charmm family of force fields see Bernard Brooks's Charmm home page or the MacKerrel web site or the Charmm.org web site.. Note that Charmm22 is a Lennard-Jones (12-6) force field and can only be combined with other Lennard-Jones (12-6) force fields. I would like to acknowledge Alex D. MacKerrel and Tom B. Woolf for providing useful guidance about implementing Charmm22. Any discrepencies (especially typos) from the published Charmm22 force field values are the sole responsibility of Marcus G. Martin, and I welcome feedback on how this implementation compares with other programs.

References for Charmm22

The best literature reference for Charmm22 is

- MacKerell *et al.* 1998

An electronic version of the Charmm22 parameters can be found at the MacKerell force fields site.

Charmm22 in Towhee

The official force field name for Charmm22 in Towhee is 'Charmm22'. Here I list all of the Charmm22 atom names for use in the towhee_input file, along with a brief description taken from the MacKerell *et al.* 1998 paper supplementary information. I have added some comments where I thought clarification was needed, these are all in [square brackets]. Please note that the capitalization and spacing pattern is important and must be followed exactly as listed here.

- **'C'** : polar C
- **'CA'** : aromatic C
- **'CT1'** : aliphatic sp^3 C for CH
- **'CT2'** : aliphatic sp^3 C for CH_2
- **'CT3'** : aliphatic sp^3 C for CH_3
- **'CPH1'** : his CG and CD2 carbons
- **'CPH2'** : his CE1 carbon
- **'CPT'** : trp C between rings
- **'CY'** : trp C in pyrrole ring
- **'CP1'** : tetrahedral C (proline CA)
- **'CP2'** : tetrahedral C (proline CB/CG)
- **'CP3'** : tetrahedral C (proline CD)
- **'CC'** : carbonyl C for sidechains asn, asp, gln, glu
- **'CD'** : carbonyl C for none amides, asp, glu, cter
- **'CPA'** : heme alpha-C
- **'CPB'** : heme beta-C
- **'CPM'** : heme meso-C
- **'CM'** : heme CO carbon
- **'CS'** : thiolate carbon
- **'CE1'** : for alkene; RHC=CR
- **'CE2'** : for alkene; H_2C=CR

- **'FE'** : heme iron
- **'H'** : polar H
- **'HC'** : N-ter H
- **'HA'** : nonpolar H
- **'HT'** : TIPS3P water hydrogen
- **'HP'** : aromatic H
- **'HB'** : backbone H
- **'HR1'** : his he1, (+) his HG,HD2
- **'HR2'** : (+) his HE1
- **'HR3'** : neutral his HG, HD2
- **'HS'** : thiol hydrogen
- **'HA1'** : for alkene; RHC=CR
- **'HA2'** : for alkene; H_2C=CR
- **'N'** : proline N
- **'NR1'** : neutral his protonated ring nitrogen
- **'NR2'** : neutral his unprotonated ring nitrogen
- **'NR3'** : charged his ring nitrogen
- **'NH1'** : peptide nitrogen
- **'NH2'** : amide nitrogen
- **'NH3'** : ammonium nitrogen
- **'NC2'** : guanidinium nitrogen
- **'NY'** : trp N in pyrrole ring
- **'NP'** : proline ring NH_2^+ (N-terminal)
- **'NPH'** : heme pyrrole N
- **'O'** : carbonyl oxygen
- **'OB'** : carbonyl oxygen in acetic acid
- **'OC'** : carboxylate oxygen
- **'OH1'** : hydroxyl oxygen
- **'OS'** : ester oxygen
- **'OT'** : TIPS3P water oxygen
- **'OM'** : heme CO/O2 oxygen
- **'S'** : sulphur
- **'SM'** : sulfur C-S-S-C type
- **'SS'** : thiolate sulfur

Coulombic Interactions

Charmm 22 utilizes point charges on atomic centers to represent the charge distribution on a molecule. As far as I know, there is no automated system for assigning the charges in Charmm 22. However, Charmm22 uses a neutral group aproach for most moities found in organic molecules. It is fairly easy to scan through the example molecular charge distributions found in the files available at the MacKerell research web site and determine what charges to apply to the molecule you wish to simulate. His Charmm Empirical Force Fields web page is especially helpful.

Improper torsions

Improper torsions are not automatically generated by the Towhee code as the rules for determining where they are applied are not always straight-forward. Charmm22 exclusively uses the out-of-plane version of the improper torsions, and they are typically centered on an sp^2 atom in order to enforce planarity with its three neighbors. These torsions are listed in the Charmm22 literature as i-j-k-l where the angle is the dihedral between i-j-k and j-k-l. The bonding pattern is not completely clear to me, but it appears that either i or l is the central atom which is bonded to all three of the other atoms, and none of the other three atoms are bonded to each other. In the towhee_input file the improper torsions is listed starting from the central atom and the three other atoms are listed in the same order as Charmm22. So, if the central atom is i, then the atoms are listed j, k, l. If the central atom is l then the atoms are listed k, j, i. Remember that you can set the improper type to 0 to have the code automatically determine the improper type (so long as inpstyle is 2).

Proteins

All of the 20 basic amino acids (including the 3 forms of hystidine and the thiol and disulfide forms of cysteine) are functional for the Charmm22 force field. I have implemented the atom types and charges according to the published Charmm22 values. Below is a complete list of the codes for the 20 amino acids, plus some other functional groups that work with the protein builder.

- 'a0' alanine
- 'c0' cysteine with hydrogen on the sulfur
- 'cs' cysteine in a disulfide bond
- 'cp' cysteine bonded to palmitate (includes the palmitate)
- 'd-' aspartic acid deprotonated
- 'e-' glutamic acid deprotonated
- 'f0' phenylalanine
- 'g0' glycine
- 'h+' histidine both N protonated
- 'hd' histidine neutral with only N_d protonated
- 'he' histidine neutral with only N_e protonated
- 'i0' isoleucine
- 'k+' lysine protonated
- 'kr' lysine-like retinal
- 'l0' leucine
- 'm0' methionine
- 'n0' asparagine
- 'p0' proline
- 'q0' glutamine
- 'r+' arginine protonated
- 's0' serine
- 't0' threonine
- 'v0' valine
- 'w0' tryptophan
- 'y0' tyrosine
- 'za' acetyl cap on the N-terminus
- 'ze' ethanolamine cap on the C-terminus
- 'zf' formaldahyde cap on the N-terminus
- 'zm' amide cap (NH_2) on the C-terminus. Note: differs from official charmm22 by having a 0.31 charge on both terminal H instead of giving one 0.30 and the other 0.32.
- 'zn' N-methylamide cap on the C-terminus

Return to the Towhee Capabilities web page

Send comments to: Marcus G. Martin
Last updated: May 12, 2006

MCCCS Towhee (Charmm22 Extra Parameters for Fluoroalkanes)

Overview

This section covers some extra parameters for use in combination with the CHARMM22 force field as it is implemented into the towhee_ff_Charmm22fe file in the ForceFields directory. This file is only likely to be useful in combination with the standard CHARMM22 forcefield. Charmm22x is a Lennard-Jones (12-6) force field and typically uses the Lorentz-Berthelot mixing rules. The parameters in this force field were developed for fluoroethanes. However, to extend the force field's capability to handle other fluorohydrocarbons, missing bond, angle and torsion terms are assumed to be equal to the similar ones reported in the reference. These new additions are the sole responsibility of Marcus G. Martin.

References for Charmm22fe

The references for these additional parameters

- Chen et. al 2001
  and MacKerell force fields site.

Charmm22fe in Towhee

The official force field name for Charmm222fe in Towhee is 'Charmm22' as the name has to be the same in order to get the molecule assembler to consider these parameters when building a molecule. Here is the list of atom names for use in the towhee_input file, along with a brief description. Please note that the capitalization and spacing pattern is important and must be followed exactly as listed here.

  - **'CF1'** : carbon in monofluoroethane
  - **'CF2'** : carbon in difluoroethane
  - **'CF3'** : carbon in trifluoroethane
  - **'HF1'** : hydrogen in monofluoroethanes
  - **'HF2'** : hydrogen in difluoroethanes
  - **'F1'** : fluorine in monofluoroethane
  - **'F2'** : fluorine in difluoroethane
  - **'F3'** fluorine in trifluoroethane

Coulombic interactions

In the original paper following partial charges were reported

- **'CF1'** -0.06
- **'CF2'** 0.24
- **'CF3'** 0.38
- **'HF1'** 0.11
- **'HF2'** 0.10
- **'F1'** -0.22
- **'F2'** -0.19
- **'F3'** -0.15

Return to the Towhee Capabilities web page

MCCCS Towhee (Charmm27)

Overview

This section covers the Charmm27 force field as it is implemented into the towhee_ff_Charmm27 file in the ForceFields directory. All of the Towhee atom types for the Charmm27 force field are listed, along with a short description of their meanings. Charmm27 is a Lennard-Jones (12-6) force field and typically uses the Lorentz-Berthelot mixing rules. Any discrepencies (especially typos) from the published Charmm27 force field values are the sole responsibility of Marcus G. Martin, and I welcome feedback on how this implementation compares with other programs.

References for Charmm27

The best literature references for the Charmm27 forcefield parameters are

- Foloppe and MacKerell 2000

and

- MacKerell *et al.* 1998.

There are parameter (par_all27_prot_na.inp) and topology (top_all27_prot_na.inp) files that are extremely helpful, and can be found on Alex MacKerell's force field web site. There is also a forum for Charmm discussion at Charmm.org.

Charmm27 in Towhee

The official force field name for Charmm27 in Towhee is 'Charmm27'. Here I list all of the Charmm27 atom names for use in the towhee_input file, along with a brief description. Please note that the capitalization and spacing pattern is important and must be followed exactly as listed here.

Calcium

- **'CAL'** : Calcium ion

Carbon

- **'C'** : Carbonyl carbon, peptide backbone
- **'CA'** : Aromatic carbon
- **'CC'** : Carbonyl carbon (in Asn, Asp, Gln, Glu, and C terminal)
- **'CD'** : Carbonyl carbon (in protonated Glu and Asp)
- **'CE1'** : Carbon in alkene (R-CH=CH-R) non-lipid version. This is essentially the same as the **CEL1** term as Charmm27 has some redundancy in its naming conventions.
- **'CE2'** : Carbon in ethene (CH2=CH2) non-lipid version. This is essentially the same as the **CEL2** term as Charmm27 has some redundancy in its naming conventions.
- **'CEL1'** : Carbon in alkene (R-CH=CH-R) lipid. This is essentially the same as the **CE1** term as Charmm27 has some redundancy in its naming conventions.
- **'CEL2'** : Carbon in ethene (CH2=CH2) lipid version. This is essentially the same as the **CE2** term as Charmm27 has some redundancy in its naming conventions.
- **'CL'** : Methyl Acetate update.
- **'CM'** : Heme CO carbon
- **'CN1'** : Carbonyl carbon (Nucleic Acids)
- **'CN1A'** : NAD+/NADH amide carbonyl carbon (Nucleic Acids)
- **'CN1T'** : Carbonyl carbon for thymine/uridine C2 (Nucleic Acids)
- **'CN2'** : Aromatic carbon to amide (Nucleic Acids)
- **'CN3'** : Aromatic carbon (Nucleic Acids)
- **'CN3A'** : Aromatic carbon for NAD+ (Nucleic Acids)
- **'CN3B'** : Aromatic carbon for NAD+ (Nucleic Acids)

'CN3C' : Aromatic carbon for NADH (Nucleic Acids)

- **'CN3D'** : Aromatic carbon for 5-methylcytosine (Nucleic Acids)
- **'CN3T'** : Aromatic carbon for thymine C5 (Nucleic Acids)
- **'CN4'** : Carbon for purine C8 and ADE C2 (Nucleic Acids)
- **'CN5'** : Carbon for purine C4 and C5 (Nucleic Acids)
- **'CN5G'** : Carbon for quanine C5 (Nucleic Acids)
- **'CN7'** : Aliphatic sp3 carbon for CH (equivalent to protein CT1) (Nucleic Acids)
- **'CN7B'** : Aliphatic carbon for C1' (Nucleic Acids)
- **'CN7C'** : Carbon for C2' in arabinose (Nucleic Acids)
- **'CN7D'** : Carbon for C2' in nucleic acid fluorine derivatives (Nucleic Acids)
- **'CN8'** : Aliphatic sp3 carbon for CH2 (equivalent to protein CT2) (Nucleic Acids)
- **'CN8B'** : Aliphatic sp3 carbon for CH2 (equivalent to protein CT2) (Nucleic Acids)
- **'CN9'** : Aliphatic sp3 carbon for CH3 (equivalent to protein CT3) (Nucleic Acids)
- **'CNA'** : Pure aromatic carbon (Nucleic Acids)
- **'CNA2'** : Pure aromatic carbon bound to fluorine (Nucleic Acids)
- **'CNE1'** : Carbon in alkene (RHC=CR) (Nucleic Acids)
- **'CNE2'** : Carbon in alkene (H2C=CR) (Nucleic Acids)
- **'CP1'** : Tetrahedral carbon (proline CA)
- **'CP2'** : Tetrahedral carbon (proline CB/CG)
- **'CP3'** : Tetrahedral carbon (proline CD)
- **'CPA'** : Heme alpha-carbon
- **'CPB'** : Heme beta-carbon
- **'CPH1'** : Histidine CG and CD2 carbons
- **'CPH2'** : Histidine CE1 carbon
- **'CPM'** : Heme meso-carbon
- **'CPT'** : Tryptophan carbon between rings
- **'CS'** : Thiolate carbon
- **'CT1'** : Aliphatic sp^3 carbon for CH non-lipid version
- **'CT2'** : Aliphatic sp^3 carbon for CH2 non-lipid version
- **'CT3'** : Aliphatic sp^3 carbon for CH3 non-lipid version
- **'CTL1'** : Aliphatic sp^3 carbon for CH lipid version
- **'CTL2'** : Aliphatic sp^3 carbon for CH2 lipid version
- **'CTL3'** : Aliphatic sp^3 carbon for CH3 lipid version
- **'CTL5'** : tetramethylammonium
- **'CY'** : Tryptophan carbon in pyrole ring

Cesium

- **'CES'** : Cesium ion

Chlorine

- **'CLA'** : Chlorine ion

Fluorine

- **'FA1'** : Aromatic fluorine
- **'FN1'** : Fluorine for sugar derivatives

Hydrogen

- **'H'** : Polar hydrogen
- **'HA'** : Nonpolar hydrogen
- **'HAL1'** : Nonpolar hydrogen in a lipid, bonded to **CTL1**
- **'HAL2'** : Nonpolar hydrogen in a lipid, bonded to **CTL2**
- **'HAL3'** : Nonpolar hydrogen in a lipid, bonded to **CTL3**
- **'HB'** : Backbone hydrogen
- **'HC'** : Hydrogen in N terminal group
- **'HCL'** : Hydrogen in ethanolamine
- **'HE1'** : Hydrogen in alkene (RHC=CR)

'HE2' : Hydrogen in alkene (H2C=CR)

- **'HEL1'** : Hydrogen in alkene (lipid version) bonded to **CEL1**
- **'HEL2'** : Hydrogen in alkene (lipid version) bonded to **CEL2**
- **'HL'** : Polar hydrogen on NC4+
- **'HN1'** : Amine proton (Nucleic Acids)
- **'HN2'** : Ring nitrogen proton (Nucleic Acids)
- **'HN3'** : Aromatic carbon proton (Nucleic Acids)
- **'HN3B'** : NAD+ aromatic hydrogen (Nucleic Acids)
- **'HN3C'** : Standard aromatic hydrogen (as in benzene) (Nucleic Acids)
- **'HN4'** : Phosphate hydroxyl proton (Nucleic Acids)
- **'HN5'** : Ribose hydroxyl proton (Nucleic Acids)
- **'HN6'** : Ribose aliphatic proton (Nucleic Acids)
- **'HN7'** : Nonpolar hydrogen (equivalent to protein HA) (Nucleic Acids)
- **'HN8'** : Hydrogen bound to CN8 in nucleic acids/model compounds (Nucleic Acids)
- **'HN9'** : Hydrogen bound to CN9 in nucleic acids/model compounds (Nucleic Acids)
- **'HNE1'** : Hydrogen in alkene (RHC=CR) (Nucleic Acids)
- **'HNE2'** : Hydrogen in alkene (H2C=CR) (Nucleic Acids)
- **'HNP'** : Pure aromatic hydrogen (Nucleic Acids)
- **'HOL'** : Hydrogen bondd to oxygen in acetic acid
- **'HP'** : Aromatic hydrogen
- **'HR1'** : Histidine HE1 hydrogen, (+)-charged histidine HG and HD2 hydrogens
- **'HR2'** : (+)-charged histidine HE1 hydrogen
- **'HR3'** : Histidine HG and HD2 hydrogens
- **'HS'** : Thiol hydrogen
- **'HT'** : Water hydrogen, modified TIP3P model

Iron

- **'FE'** : Iron in heme

Magnesium

- **'MG'** : Magnesium ion

Nitrogen

- **'N'** : Proline nitrogen
- **'NC2'** : Guanidinium nitrogen
- **'NH1'** : Peptide nitrogen
- **'NH2'** : Amide nitrogen
- **'NH3'** : Ammonium nitrogen
- **'NH3L'** : Ethanolamine nitrogen
- **'NN1'** : Amide nitrogen (Nucleic Acids)
- **'NN1C'** : Imine nitrogen (cytosine) (Nucleic Acids)
- **'NN2'** : Protonated ring nitrogen (Nucleic Acids)
- **'NN2B'** : For N9 in guanine (Nucleic Acids)
- **'NN2C'** : Protonated ring nitrogen (cytosine) (Nucleic Acids)
- **'NN2G'** : Protonated ring nitrogen for guanine N1 (Nucleic Acids)
- **'NN2U'** : Protonated ring nitrogen for uridine N3 (Nucleic Acids)
- **'NN3'** : Unprotonated ring nitrogen (Nucleic Acids)
- **'NN3A'** : Unprotonated ring nitrogen for adenine N1 and N3 (Nucleic Acids)
- **'NN3G'** : Unprotonated ring nitrogen for guanine N3 (Nucleic Acids)
- **'NN3I'** : Unprotonated ring nitrogen for inosine N3 (Nucleic Acids)
- **'NN4'** : Nitrogen for purine N7 (Nucleic Acids)
- **'NN5'** : sp2 amine nitrogen (Nucleic Acids)
- **'NN6'** : sp3 amine nitrogen (equivalent to protein NH3) (Nucleic Acids)
- **'NP'** : Proline ring NH2+
- **'NPH'** : Heme pyrole nitrogen
- **'NR1'** : Neutral histidine protonated ring nitrogen
- **'NR2'** : Neutral histidine unprotonated ring nitrogen

- **'NR3'** : (+)-charged histidine ring nitrogen
- **'NTL'** : tetramethyl ammonium
- **'NY'** : Tryptophan nitrogen in pyrole ring

Oxygen

- **'O'** : Carbonyl oxygen
- **'O2L'** : Oxygen in sulfate or phosphate
- **'OB'** : Carbonyl oxygen in acetic acid non-lipid version
- **'OBL'** : Carbonyl oxygen in acetic acid lipid version
- **'OC'** : Carboxylate oxygen non-lipid version
- **'OCL'** : Carboxylate oxygen lipid version
- **'OH1'** : Hydroxyl oxygen non-lipid version
- **'OHL'** : Hydroxyl oxygen lipid version
- **'OM'** : Heme CO/O2 oxygen
- **'ON1'** : Carbonyl oxygen (Nucleic Acids)
- **'ON1C'** : Carbonyl oxygen for cytosine O2 (Nucleic Acids)
- **'ON2'** : Phosphate ester oxygen (Nucleic Acids)
- **'ON3'** : Double bonded oxygen in phospate (Nucleic Acids)
- **'ON4'** : Phospate hydroxyl oxygen (Nucleic Acids)
- **'ON5'** : Ribose hydroxyl oxygen (Nucleic Acids)
- **'ON6'** : Deoxyribose ring oxygen (Nucleic Acids)
- **'ON6B'** : Ribose ring oxygen (Nucleic Acids)
- **'OS'** : Ester oxygen non-lipid version
- **'OSL'** : Ester oxygen lipid version
- **'OT'** : Water oxygen, modified TIP3P model

Phosphorous

- **'P'** : Phosphorous non-lipid version
- **'P2'** : Pyrophosphate phosphorous
- **'PL'** : Phosphorous lipid version

Potassium

- **'POT'** : Potassium ion

Sodium

- **'SOD'** : Sodium ion

Sulfur

- **'S'** : Sulfur
- **'SL'** : Sulfur in sulfate
- **'SM'** : Sulfur (C-S-S-C type)
- **'SS'** : Thiolate sulfur

Coulombic Interactions

Charmm27 utilizes point charges on atomic centers to represent the charge distribution on a molecule. As far as I know, there is no automated system for assigning the charges in Charmm27. Otherwise, it is fairly easy to scan through the example molecular charge distributions found in the files available at the MacKerell research web site and determine what charges to apply to the molecule you wish to simulate.

Improper torsions

Improper torsions are not automatically generated by the Towhee code as the rules for determining where they are applied are not always straight-forward. Charmm27 exclusively uses the out-of-plane version of the improper torsions, and they are typically centered on an sp^2 atom in order to enforce planarity with its three neighbors. These torsions are listed in the Charmm27 literature as i-j-k-l where the angle is the dihedral between i-j-k and j-k-l. The bonding pattern is not completely clear to me, but it appears that either i or l is the central atom which is bonded to all three of the other atoms, and none of the other three atoms are bonded to each other. In the towhee_input file the improper torsions is listed starting from the central atom and the three other atoms are listed in the same order as Charmm27. So, if the central atom is i, then the atoms are listed j, k, l. If the central atom is l then the atoms are listed k, j, i. Remember that you can set the improper type to 0 to have the code automatically determine the improper type (so long as inpstyle is 2).

Proteins

All of the 20 basic amino acids (including the 3 forms of hystidine and the thiol and disulfide forms of cysteine) are functional for the Charmm27 force field. I have implemented the atom types and charges according to the published Charmm27 values. Below is a complete list of the codes for the 20 amino acids, plus some other functional groups that work with the protein builder.

- 'a0' alanine
- 'c0' cysteine with hydrogen on the sulfur
- 'cs' cysteine in a disulfide bond
- 'd-' aspartic acid deprotonated
- 'e-' glutamic acid deprotonated
- 'f0' phenylalanine
- 'g0' glycine
- 'h+' histidine both N protonated
- 'hd' histidine neutral with only N_d protonated
- 'he' histidine neutral with only N_e protonated
- 'i0' isoleucine
- 'k+' lysine protonated
- 'l0' leucine
- 'm0' methionine
- 'n0' asparagine
- 'p0' proline
- 'q0' glutamine
- 'r+' arginine protonated
- 's0' serine
- 't0' threonine
- 'v0' valine
- 'w0' tryptophan
- 'y0' tyrosine
- 'za' acetyl cap on the N-terminus
- 'zm' amide cap (NH_2) on the C-terminus. Note: differs from official Charmm27 by having a 0.31 charge on both terminal H instead of giving one 0.30 and the other 0.32.
- 'zn' N-methylamide cap on the C-terminus

Nucleic acids

All of the basic nucleic acids are functional for the Charmm27 force field. I have implemented the atom types and charges according to the published Charmm27 values. Below is a complete list of the codes for the nucleic acids. Notice that there is a 'd' as the first character in the codes for the deoxyribonucleotides (used in DNA) and no 'd' in the codes for the ribonucleotides (used in RNA). Generally, thymine is only found in DNA and uridine is only found in RNA, but both the deoxyribonucleotides and the ribonucleotides are implemented here. The shortname is also provided (in parenthesis) for use with the buildpartial features. .

- 'A' adenosine (AMP)
- 'dA' deoxyadenosine (dAMP)

- 'C' cytidine (CMP)
- 'dC' deoxycytidine (dCMP)
- 'G' guanosine (GUA)
- 'dG' deoxyguanosine (dGUA)
- 'T' thymidine (TMP)
- 'dT' deoxythymidine (dTMP)
- 'U' uridine (UMP)
- 'dU' deoxyuridine (dUMP)

Return to the Towhee Capabilities web page

Send comments to: Marcus G. Martin
Last updated: October 17, 2006

MCCCS Towhee (Charmm27 Extra Parameters)

Overview

This section covers some extra parameters for use in combination with the CHARMM27 force field as it is implemented into the towhee_ff_Charmm27x file in the ForceFields directory. This file is only likely to be useful in combination with the standard CHARMM27 forcefield. Charmm27x is a Lennard-Jones (12-6) force field and typically uses the Lorentz-Berthelot mixing rules. These values are not actually from the CHARMM27 forcefield and are the sole responsibility of Marcus G. Martin.

References for Charmm27x

These are some additional parameters created by Marcus G. Martin for use with the actual CHARMM27 forcefield. Some of these parameters were published in the following paper.

- Martin and Biddy 2005

Typos and comments for Charmm27x

- This file is an example of the suggested method for users to add additional parameters to existing forcefields. By placing all of the additional parameters into a separate file this makes it easy to both keep track of your changes for eventual publication, and to upgrade to new code versions without worrying about overwriting your new parameters.
- In order to use these additional parameters with the CHARMM27 forcefield you need to include both the towhee_ff_Charmm27 and towhee_ff_Charmm27x forcefield files in towhee_input. Label the molecules with the 'Charmm27' forcefield as normal and Towhee will search through the forcefield files in the order specified and take the first complete match.
- The following parameters were published in Martin and Biddy 2005.
  - CT3-CC-CT3: used for acetone and taken from the CHARMM27 CT3-CT2-CT2 parameters.

Charmm27x in Towhee

The official force field name for Charmm27x in Towhee is 'Charmm27' as the name has to be the same in order to get the molecule assembler to consider these parameters when building a molecule. There are no additional nonbonded types in this forcefield. See the original CHARMM27 forcefield for the valid atom types.

Return to the Towhee Capabilities web page

Send comments to: Marcus G. Martin
Last updated: May 03, 2006

MCCCS Towhee (Charmm27 Rigid Water)

Overview

This section covers the Charmm27 Rigid Water force field as it is implemented into the towhee_ff_C27rigid file in the ForceFields directory. All of the Towhee atom types for this force field are listed, along with a short description of their meanings. This is a Lennard-Jones (12-6) force field and typically uses the Lorentz-Berthelot mixing rules. It uses the same parameters as the regular Charmm27 force field, with the notable exception that all of the intramolecular terms are rigid. Any discrepencies (especially typos) from the published force field values are the sole responsibility of Marcus G. Martin, and I welcome feedback on how this implementation compares with other programs.

References for Charmm27 Rigid Water

The best literature references for the Charmm27 forcefield parameters are

- Foloppe and MacKerell 2000

and

- MacKerell *et al.* 1998.

More information about the Charmm27 forcefield is available on the Towhee Charmm27 web manual page. C27rigid is simply a rigid implementation of the Charmm27 water model where the equilibrium values of the bond and angle are set to be rigid instead of flexible. This force field was implemented to allow comparison with simulations using the standard molecular dynamics practice of freezing out intramolecular motion in water in order to speed the simuluations.

C27rigid in Towhee

The official force field name for Charmm27 Rigid Water in Towhee is 'C27rigid'. Here I list all of the atom names for use in the towhee_input file, along with a brief description. Please note that the capitalization and spacing pattern is important and must be followed exactly as listed here.

Hydrogen

- **'HT'** : Water hydrogen, modified TIP3P model

Oxygen

- **'OT'** : Water oxygen, modified TIP3P model

Coulombic Interactions

Charmm27 Rigid Water utilizes point charges on atomic centers to represent the charge distribution on a molecule. The 'bond increment' method for assigning charges is implemented for this force field. Please see the inpstyle 2 documentation for further information on enabling this option. Here I list the bond increments for the atom names listed above. The value of the bond increment is added to the first atom listed and subtracted from the second atom listed.

- HT - OT: 0.417

Return to the Towhee Capabilities web page

Send comments to: Marcus G. Martin

Last updated: September 07, 2005

MCCCS Towhee (ClayFF)

Overview

This section covers the ClayFF force field as it is implemented into the towhee_ff_ClayFF file in the ForceFields directory. All of the Towhee atom types for the ClayFF force field are listed, along with a short description of their meanings. Any discrepencies (especially typos) from the published force field values are the sole responsibility of Marcus G. Martin, and I welcome feedback on how this implementation compares with other programs.

References for ClayFF

There is currently one source for ClayFF.

- Cygan *et al.* 2004

ClayFF in Towhee

The official force field name for these parameters is 'ClayFF'. Here I list all of the ClayFF atom names for use in the towhee_input file, along with a brief description. Please note that the capitalization and spacing pattern is important and must be followed exactly as listed here.

- **'h*'** : water hydrogen
- **'ho'** : hydroxyl hydrogen
- **'o*'** : water oxygen
- **'oh'** : hydroxyl oxygen
- **'ob'** : bridging oxygen
- **'obos'** : bridging oxygen with octahedral substitution
- **'obts'** : bridging oxygen with tetrahedral substitution
- **'obss'** : bridging oxygen with double substitution
- **'ohs'** : hydroxyl oxygen with substitution
- **'st'** : tetrahedral silicon
- **'ao'** : octahedral aluminum
- **'at'** : tetrahedral aluminum
- **'mgo'** : octahedral magnesium
- **'mgh'** : hydroxide magnesium
- **'cao'** : octahedral calcium
- **'cah'** : hydroxide calcium
- **'feo'** : octahedral iron
- **'lio'** : octahedral lithium
- **'Na'** : aqueous sodium ion
- **'K'** : aqueous potassium ion
- **'Cs'** : aqueous cesium ion
- **'Ca'** : aqueous calcium ion
- **'Ba'** : aqueous barium ion
- **'Cl'** : aqueous chloride ion

Coulombic Interactions

This force field uses point charges on atomic centers. Here we list suggested charges for each of the atom types in this force field.

- **'h*'** : 0.4100
- **'ho'** : 0.4250
- **'o*'** : -0.8200

- **'oh'** : -0.9500
- **'ob'** : -1.0500
- **'obos'** : -1.1808
- **'obts'** : -1.1688
- **'obss'** : -1.2996
- **'ohs'** : -1.0808
- **'st'** : 2.1000
- **'ao'** : 1.5750
- **'at'** : 1.5750
- **'mgo'** : 1.3600
- **'mgh'** : 1.0500
- **'cao'** : 1.3600
- **'cah'** : 1.0500
- **'feo'** : 1.5750
- **'lio'** : 0.5250
- **'Na'** : 1.0
- **'K'** : 1.0
- **'Cs'** : 1.0
- **'Ca'** : 2.0
- **'Ba'** : 2.0
- **'Cl'** : -1.0

Improper torsions

There are no improper torsions in this force field.

Proteins

This forcefield is not designed for proteins.

Return to the Towhee Capabilities web page

Send comments to: Marcus G. Martin
Last updated: September 07, 2005

MCCCS Towhee (Coon *et al.* 1987)

Overview

This section covers the Coon *et al.* (Coon1987) force field as it is implemented into the towhee_ff_Coon1987 file in the ForceFields directory. This force field was originally designed for fluid simulations of O_2 and N_2. Note that this is a Lennard-Jones force field that uses Lorentz-Berthelot mixing rules so it is easily combined with other force fields. Any discrepencies (especially typos) from the published values are the sole responsibility of Marcus G. Martin, and I welcome feedback on how this implementation compares with other programs.

References for Coon1987

What I term Coon1987 is described in the following paper.

- Coon *et al.* 1987

Coon1987 in Towhee

The official force field name for Coon *et al.* 1987 in Towhee is 'Coon1987'. Here I list all of the atom names for use in the towhee_input file, along with a brief description. I created these atom names in order to work with the molecule assembler. Please note that the capitalization and spacing pattern is important and must be followed exactly as listed here.

- **'O'** : Oxygen bonded to another oxygen in O_2
- **'N'** : Nitrogen bonded to another nitrogen in N_2

Coulombic interactions

This force field does not assign any coulombic interactions to the atoms. The 'bond increment' method of **charge_assignment** is implemented for this forcefield, but it just assigns zero charge to all of the atoms. Please see the inpstyle 2 documentation for further information on enabling this option. Otherwise, you are welcome to manually set the charges. Here I list the charges that are assigned by the 'bond increment' method.

- **'O'** 0.0
- **'N'** 0.0

Return to the Towhee Capabilities web page

Send comments to: Marcus G. Martin
Last updated: September 07, 2005

MCCCS Towhee (DACNIS United Atom)

Overview

This section covers the DACNIS United Atom force field as it is implemented into the towhee_ff_DACNIS-UA file in the ForceFields directory. All of the Towhee atom types for the DACNIS United Atom force field are listed, along with a short description of their meanings. DACNIS-UA is a little force field I created for performing some simulations of methane and ethane adsorption in silicalite using a cut-and-shift potential. This was useful as the main application of the force field was computing the transport of methane through silicalite using the dual control volume grand canoical molecular dynamics code Ladera. Note that DACNIS United Atom is a Lennard-Jones (12-6) force field and can only be combined with other Lennard-Jones (12-6) force fields. I welcome feedback on how this implementation compares with other programs (assuming anyone else has ever bothered to implement this force field).

References for DACNIS-UA

The literature reference for DACNIS-UA is

- Martin *et al.* 2001

DACNIS-UA in Towhee

The official force field name for DACNIS United Atom in Towhee is 'DACNIS-UA'. Here I list all of the DACNIS atom names for use in the towhee_input file, along with a brief description. I created these atom names in order to work with the molecule assembler. DACNIS-UA is a united atom force field which lumps atoms together. There are only three different groups at the moment. Note that the Ozeo group is designed for a fixed conformation of silicalite so it has no bonded interactions. Please note that the capitalization and spacing pattern is important and must be followed exactly as listed here.

- **'CH4sp3'** : methane
- **'CH3sp3'** : sp^3 non-aromatic carbon bonded to three hydrogens and one heavy atom
- **'Ozeo'** : oxygen groups in silicalite.

Coulombic interactions

There are currently no coulombic interactions in the DACNIS-UA force field.

Improper torsions

DACNIS-UA does not utilize improper torsions.

Proteins

DACNIS-UA does not have all of the groups needed for protein simulations.

Return to the Towhee Capabilities web page

Send comments to: Marcus G. Martin
Last updated: September 07, 2005

MCCCS Towhee (DREIDING)

Overview

This section covers the DREIDING Force Field as it is implemented into the towhee_ff_DREIDING file in the ForceFields directory. All of the Towhee atom types for this force field are listed, along with a short description of their meanings. Note that this is a Lennard-Jones force field and is easily combined with other similar force fields. You need to use the classical_potential 'Lennard-Jones' for the this force field and the suggested mixing rules are 'Geometric', although 'Lorentz-Berthelot' is also enabled. Any discrepencies (especially typos) from the published force field values are the sole responsibility of Marcus G. Martin, and we welcome feedback on how this implementation compares with other programs.

References for DREIDING

The parameters implemented here are from Tables I, II, and III of the primary DREIDING paper.

- Mayo *et al.* 1990

The special hydrogen bonding interactions are not implememted and therefore the atom type 'H___HB' is not included.

DREIDING in Towhee

The official force field name for DREIDING in Towhee is 'DREIDING'. Here is a list of the atom names for use in the towhee_input file, along with a brief description taken from the DREIDING literature. DREIDING uses a five-character label to describe every element. The first two letters are the chemical symbol (appended with an underscore for single letter elements). The third character describes the geometry of the molecule as follows.

- 1: linear
- 2: trigonal
- R: resonant
- 3: tetrahedral

The fourth and fifth characters are there to help distinguish between otherwise similar atoms (for example, the charge state of metals and special characters for certain atoms). Towhee follows the DREDING naming convension. Please note that the atom type 'H__HB' is omitted from this implementation as we did not wish to deal with the special hydrogen bonding interactions that have subsequently fallen out of favor with most force field developers. The element names are generally obvious (given the rules above), but a notes are added to some potentially confusing elements, and to the united-atom options. Please note that the capitalization and spacing pattern is important and must be followed exactly as listed here.

All Atom

- **'H_'**
- **'H_b'**: hydrogen bridging between two boron atoms
- **'B_3'**
- **'B_2'**
- **'C_3'**
- **'C_R'**
- **'C_2'**
- **'C_1'**
- **'N_3'**
- **'N_R'**
- **'N_2'**
- **'N_1'**
- **'O_3'**

- **'O_R'**
- **'O_2'**
- **'O_1'**
- **'F_'**
- **'Na_+'**
- **'Al3'**
- **'Si3'**
- **'P_3'**
- **'S_3'**
- **'Cl'**
- **'Ca_+2'**
- **'Fe_+2'**
- **'Zn_+2'**
- **'Ga3'**
- **'Ge3'**
- **'As3'**
- **'Se3'**
- **'Br'**
- **'In3'**
- **'Sn3'**
- **'Sb3'**
- **'Te3'**
- **'I_'**
  United Atom lumping of Hydrogens onto Carbon
- **'C_R1'**aromatic carbon plus one bonded hydrogen
- **'C_34'**sp^3 carbon plus four bonded hydrogen (methane).
- **'C_33'**sp^3 carbon plus three bonded hydrogen (methyl group).
- **'C_32'**sp^3 carbon plus two bonded hydrogen (methylene group).
- **'C_31'**sp^3 carbon plus one bonded hydrogen (methine group).

Coulombic interactions

The DREIDING paper suggests that the user either ignore charges completely, or assign them using the method of Gasteiger and Marsili 1980. Unfortunately, this method is not currently available in Towhee

Improper torsions

DREIDING uses an improper torsion (called an inversion in their paper) on any atom (I) that is bonded to exactly three other atoms (J,K, and L) and also has a planar geometry (sp^2 or resonant central atoms). The improper considers the angle each of the vectors (IJ, IK or IL) makes with a plane described by the other substituants. For example, the angle between the IJ vector and the IKL plane. Towhee currently requires the user to specify all improper torsions, but you may toggle the type to 0 to allow automatical determination of the appropriate parameters for each improper torsion.

Return to the Towhee Capabilities web page

Send comments to: Marcus G. Martin

Last updated: December 06, 2006

MCCCS Towhee (Dubbeldam *et al.* 2004)

Overview

This section covers the Dubbeldam *et al.* (Dubb2004) force field as it is implemented into the towhee_ff_Dubb2004 file in the ForceFields directory. This force field is for adsorption simulations of alkanes in various Zeolites. Note that this is a Lennard-Jones force field that uses explicit mixing rules so it is not easily combined with other force fields. Any discrepencies (especially typos) from the published values are the sole responsibility of Marcus G. Martin, and I welcome feedback on how this implementation compares with other programs.

References for Dubb2004

What I term Dubb2004 is described in several papers.

- Dubbeldam *et al.* 2004 PRL
- Dubbeldam *et al.* 2004 JPCB
- Calero *et al.* 2004

In addition, some of their torsional interactions are overly complicated. The torsions involving the **CH** group are specified in terms of where the hydrogen would be if it was explicitly included in the simulation. Thijs Vlugt suggested replacing the torsions involving the **CH** group with those from the TraPPE-UA forcefield and I have followed that suggestion for this implementation.

Dubb2004 in Towhee

The official force field name for Dubbeldam *et al.* United Atom in Towhee is 'Dubb2004'. Here I list all of the atom names for use in the towhee_input file, along with a brief description. I created these atom names in order to work with the molecule assembler. Dubb2004 is a united-atom force field which lumps hydrogens onto their neighboring carbons. It is also designed for work with Silaceous and aluminum substituted zeolites. The general pattern of the names is a list of the atoms that make up the group, followed by the hybridization of the central atom, and ending in a character string that distinguishes similar atoms. Please note that the capitalization and spacing pattern is important and must be followed exactly as listed here.

sp^3 hybridized Carbons

- **'CH4'** : sp^3 carbon plus the 4 hydrogens bonded to it (united-atom) in methane.
- **'CH3'** : sp^3 carbon plus the 3 hydrogens bonded to it (united-atom) in ethane.
- **'CH2'** : sp^3 carbon plus the 2 hydrogens bonded to it (united-atom).
- **'CH'** : sp^3 carbon plus the 1 hydrogen bonded to it (united-atom).
- **'C'** : sp^3 carbon that is not bonded to any hydrogens.

Zeolite Framework Atoms

- **'O'** : oxygen bonded to silicon and/or aluminum in the zeolite framework.
- **'Si'** : silicon bonded to four oxygens in the zeolite framework.
- **'Al'** : aluminum bonded to four oxygens in the zeolite framework.

Zeolite Counter Ions

- **'Na'** : sodium ion.

Coulombic interactions

There are specific charge assignents for the atoms in the zeolite framework and these assignments are enforced by the angle potential for the zeolite. There are no charges on the alkane united-atoms. The 'bond increment' method of **charge_assignment** is implemented for this forcefield. Please see the inpstyle 2 documentation for further information on enabling this option. Otherwise, you are welcome to manually set the charges. Here I list the charges that are assigned by the 'bond increment' method.

- **'Si'** : 2.05
- **'Al'** : 1.75
- **'O'** bonded to two silicons: -1.025
- **'O'** bonded to one silicon and one aluminum: -1.20
- **'Na'** : 1.0
- **'C'** : all alkane carbons have a 0.0 charge.

Return to the Towhee Capabilities web page

Send comments to: Marcus G. Martin
Last updated: September 07, 2005

MCCCS Towhee (Elementary Physical Models)

Overview

This section covers three Elementary Physical Models (EPM) for CO_2 as they are implemented into the towhee_ff_EPM file in the ForceFields directory. All of the Towhee atom types for these EPM force fields are listed, along with a short description of their meanings. Note that EPM are Lennard-Jones (12-6) force fields that use Geometric mixing rules and can only be combined with other Lennard-Jones (12-6) force fields. Any discrepencies (especially typos) from the published force field values are the sole responsibility of Marcus G. Martin, and I welcome feedback on how this implementation compares with other programs.

References for EPM

The three EPM models are presented in a single paper by Harris and Yung.

- Harris and Yung 1995

EPM in Towhee

The official force field name for the Elementary Physical Models in Towhee is 'EPM'. Here I list all of the EPM atom names for use in the towhee_input file, along with a brief description. I created these atom names in order to work with the molecule assembler. There are three different variants of the carbon dioxide potentials present in this force field and all are implemented here. Please note that the capitalization and spacing pattern is important and must be followed exactly as listed here.

EPM Rigid Carbon Dioxide

This model is taken from the first part of Table 1 in Harris and Yung 1995. It has a rigid bending angle for CO_2

- **'C_EPM-R'** : central carbon in carbon dioxide.
- **'O_EPM-R'** : oxygen in carbon dioxide.

EPM Flexible Carbon Dioxide

This model is taken from the first part of Table 1 in Harris and Yung 1995. It has a flexible bending angle for CO_2

- **'C_EPM-F'** : central carbon in carbon dioxide.
- **'O_EPM-F'** : oxygen in carbon dioxide.

EPM2 Carbon Dioxide

This model is taken from the second part of Table 1 in Harris and Yung 1995. It has a flexible bending angle for CO_2

- **'C_EPM2'** : central carbon in carbon dioxide.
- **'O_EPM2'** : oxygen in carbon dioxide.

Coulombic interactions

EPM uses atom-centered point charges to represent the electrostatic interactions. The 'bond increment' method for assigning charges is implemented for these molecules. Please see the inpstyle 2 documentation for further information on enabling this option. Here I list the bond increments for the atom names listed above. The value of the bond increment is added to the first atom listed and subtracted from the second atom listed.

- C_EPM-R - O_EPM-R: 0.33225
- C_EPM-F - O_EPM-F: 0.33225
- C_EPM2 - O_EPM2: 0.3256

Return to the Towhee Capabilities web page

MCCCS Towhee (Galassi and Tildesley 1994)

Overview

This section covers the Galassi and Tildesly 1994 (Gala1994) force field as it is implemented into the towhee_ff_Gala1994 file in the ForceFields directory. This force field was designed to reproduce the vapor-liquid coexistence curves of N_2, F_2, and Cl_2. Note that this is a Lennard-Jones force field that uses Lorentz-Berthelot mixing rules so it is easily combined with other force fields. Any discrepencies (especially typos) from the published values are the sole responsibility of Marcus G. Martin, and I welcome feedback on how this implementation compares with other programs.

References for Gala1994

What I term Gala1994 is described in the following paper.

- Galassi and Tildesley 1994

Gala1994 in Towhee

The official force field name for Galassi and Tildesley 1994 in Towhee is 'Gala1994'. Here I list all of the atom names for use in the towhee_input file, along with a brief description. I created these atom names in order to work with the molecule assembler. Please note that the capitalization and spacing pattern is important and must be followed exactly as listed here.

- **'Cl'** : Chlorine bonded to another Chlorine in Cl_2
- **'F'** : Fluorine bonded to another Fluorine in F_2
- **'N'** : Nitrogen bonded to another nitrogen in N_2

Coulombic interactions

This force field does not assign any coulombic interactions to the atoms. The 'bond increment' method of **charge_assignment** is implemented for this forcefield, but it just assigns zero charge to all of the atoms. Please see the inpstyle 2 documentation for further information on enabling this option. Otherwise, you are welcome to manually set the charges. Here I list the charges that are assigned by the 'bond increment' method.

- **'Cl'** 0.0
- **'F'** 0.0
- **'N'** 0.0

Return to the Towhee Capabilities web page

Send comments to: Marcus G. Martin
Last updated: September 07, 2005

MCCCS Towhee (Gromos 43A1)

Overview

This section covers the Gromos 43A1 force field as it is implemented into the towhee_ff_Gromos43A1 file in the ForceFields directory. All of the Towhee atom, bond angle, and dihedral potential numbers for the Gromos 43A1 force field are listed, along with a short description of their meanings. Unlike almost all of the other force fields implemented into Towhee, Gromos 43A1 cannot be used with the molecule assembler (inpstyle 2) as they have not presented their parameters in a way that facilitates such an assembler, and I have not yet figured out how to create a set of atom types that would work with the assembler. One of the main barriers to such an assembler is that Gromos 43A1 does not apply the same torsional potential to all dihedrals that are connected by the same atom types. Instead Gromos 43A1 applies the torsional potential to only one of the torsions across the same two central atoms of a dihedral, and the rest interact only via 1-4 van der Waals and coulombic terms. For more information about the Gromos force field see the Gromos web site. Note that Gromos 43A1 is a Lennard-Jones (12-6) force field, but it has a rather complex algorithm for computing the cross interactions and so its parameters are listed explicitly (starting with Towhee Version 4.4.0) and so you cannot mix it with any of the other force fields in Towhee. I would like to acknowledge Lukas D. Schuler for providing very useful guidance about implementing Gromos 43A1, and a copy of the relevent pages of the Gromos manual. Any discrepencies (especially typos) from the published Gromos 43A1 force field values are the sole responsibility of Marcus G. Martin, and I welcome feedback on how this implementation compares with other programs.

References for Gromos 43A1

The literature reference for Gromos 43A1 is

- Scott *et al.* 1999

Unfortunately this reference does not actually contain the Gromos 43A1 parameters. In order to get the parameters you need a copy of the Gromos users manual. It appears that the only way to get the users manual is to purchase the Gromos code, but here is the reference for the users manual in case you want to give it a try.

- van Gunsteren *et al.* 1996

A copy of the portions of the users manual related to the force field was generously provided to me by Lukas D. Schuler.

Gromos 43A1 Towhee

The official force field name for Gromos 43A1 in Towhee is 'Gromos43A1', although that will not get you very far as the molecule assembler does not yet work for Gromos43A1. In order to make the use of Gromos 43A1 as convienent as possible without the use of the molecule builder, I have numbered the atoms, vibrations, bending angles, dihedral angles, and improper torsions using (almost) the same numbering conventions as Gromos 43A1. You will be forced to use inpstyle 0, but hopefully these lists will make the task possible for those who really wish to use Gromos 43A1. The descriptions below include the comments from the Gromos users manual, with any additional comments I felt useful in [square brackets].

Atom numbers : **Gromos atom name** : description of the atom

- 1 : **O** : carbonyl oxygen (C=0)
- 2 : **OM** : carboxyl oxygen (CO)
- 3 : **OA** : hydroxyl, sugar or ester oxygen
- 4 : **OW** : water oxygen

- 5 : **N** : peptide nitrogen (NH)
- 6 : **NT** : terminal nitrogen (NH_2)
- 7 : **NL** : terminal nitrogen (NH_3)
- 8 : **NR** : aromatic nitrogen
- 9 : **NZ** : Arg NH (NH_2)
- 10 : **NE** : Arg NE (NH)
- 11 : **C** : bare carbon
- 12 : **CH1** : aliphatic or sugar CH group
- 13 : **CH2** : aliphatic or sugar CH_2 group
- 14 : **CH3** : aliphatic CH_3 group
- 15 : **CH4** : methane
- 16 : **CR1** : aromatic CH group
- 17 : **HC** : hydrogen bond to carbon [where the carbon is not aliphatic or you would use the appropriate united-atom CHx group]
- 18 : **H** : hydrogen not bound to carbon
- 19 : **DUM** : dummy atom [not sure what this is for, but implemented it anyway]
- 20 : **S** : sulfur
- 21 : **CU1+** : copper (charge 1+)
- 22 : **CU2+** : copper (charge 2+)
- 23 : **FE** : iron (heme)
- 24 : **ZN2+** : zinc (charge 2+)
- 25 : **MG2+** : magnesium (charge 2+)
- 26 : **CA2+** : calcium (charge 2+)
- 27 : **P** : phosphorous
- 28 : **AR** : argon
- 29 : **F** : fluorine (non-ionic)
- 30 : **CL** : chlorine (non-ionic)
- 31 : **BR** : bromine (non-ionic)
- 32 : **CMet** : CH_3 group in methanol
- 33 : **OMet** : oxygen in methanol
- 34 : **NA+** : sodium (charge 1+)
- 35 : **CL-** : chlorine (charge 1-)
- 36 : **CChl** : carbon in chloroform
- 37 : **CLChl** : chlorine in chloroform
- 38 : **HChl** : hydrogen in chloroform
- 39 : **SDmso** : sulfur in DMSO
- 40 : **CDmso** : CH_3 group in DMSO
- 41 : **ODmso** : oxygen in DMSO
- 42 : **CCl4** : carbon in carbontetrachloride
- 43 : **CLCl4** : chlorine in carbontetrachloride
- 44 : **SI** : silicon

Bond Vibration numbers : Gromos examples of the vibration

- 1 : H-OA
- 2 : H-N (all)
- 3 : HC-C
- 4 : C-O
- 5 : C-OM
- 6 : CR1-NR (6-ring)
- 7 : H-S
- 8 : C-NT,NL
- 9 : C,CR1-N,NR,CR1,C (peptide, 5-ring)
- 10 : C-N,NA,NE

- 11 : C-NR (no H)(6-ring)
- 12 : C-OA
- 13 : C-NR (heme)
- 14 : CH2-C,CR1 (6-ring)
- 15 : C,CR1-CH2,C,CR1 (6-ring)
- 16 : C,CR1,CH2-NR (6-ring)
- 17 : CHn-OA
- 18 : CHn-OM
- 19 : CHn-OA (sugar)
- 20 : CHn-N,NT,NL,NZ,NE
- 21 : CHn-NR (5-ring)
- 22 : CHn-NR (6-ring)
- 23 : O,OM-P
- 24 : O-S
- 25 : CHn-CHn (sugar)
- 26 : C,CHn-C,CHn
- 27 : OA-P
- 28 : OA-SI
- 29 : CH3-S
- 30 : CH2-S
- 31 : CH1-SI
- 32 : NR-FE
- 33 : S-S
- 34 : NR(heme)-FE
- 35 : HWat-OWat
- 36 : HChl-CChl
- 37 : CChl-CLChl
- 38 : ODmso-SDmso
- 39 : SDmso-CDmso
- 40 : CCl4-CLCl4
- 41 : HWat-HWat [I believe this is for use with shake, Towhee cannot handle cyclic 3-mers]
- 42 : HChl-CLChl
- 43 : CLChl-CLChl
- 44 : ODmso-CDmso
- 45 : CDmso-CDmso
- 46 : HMet-CMet
- 47 : CLCl4-CLCl4

Bending Angle numbers : Gromos examples of the bending angle

- 1 : NR(heme)-FE-NR(heme)
- 2 : H-S-CH2
- 3 : CH2-S-CH3
- 4 : OA-P-OA
- 5 : CH2-S-S
- 6 : NR-C-CR1 (5-ring)
- 7 : CHn-CHn-CHn,NR(6-ring)(sugar)
- 8 : CHn,OA-CHn-OA,NR(ring)(sugar)
- 9 : H-NL,NT-H ; CHn-OA-CHn(sugar)
- 10 : H-NL-C,CHn ; H-NT-CHn
- 11 : X-OA,SI-X
- 12 : CHn,C-CHn-C,CHn,OA,OM,N,NE
- 13 : OM-P-OA
- 14 : CHn-CHn-C,CHn,OA,NR,NT,NL
- 15 : CHn-CH2-S

- 16 : NR(heme)-FE-NR
- 17 : H-N-CHn
- 18 : CHn,C-C-OA,N,NT,NL
- 19 : H-NE-CH2
- 20 : CH2-N-CH1
- 21 : CH3-N-C ; CHn-C-OM
- 22 : H-NT,NZ,NE-C
- 23 : H-NT,NZ-H
- 24 : H-N-CH3,H,HC (6-ring) ; H-NT-CHn
- 25 : P,SI-OA-CHn,P
- 26 : N-C-CR1 (6-ring, no H)
- 27 : NZ-C-NZ,NE
- 28 : OM-P-OM
- 29 : O-C-CHn,C ; CH3-N-CHn
- 30 : CH1,CH2-N-C
- 31 : H-N-C
- 32 : O-C-OA,N,NT,NL ; C-NE-CH2
- 33 : FE-NR-CR1(5-ring)
- 34 : - [no description is listed about this bending angle in Gromos]
- 35 : H,HC-5-ring [H atoms bonded to an atom which is part of a 5-ring]
- 36 : X(noH)-5-ring [non-H atoms bonded to an atom which is part of a 5-ring]
- 37 : OM-C-OM
- 38 : 5,6 ring connection
- 39 : SI-OA-SI
- 40 : HWat-OWat-HWat
- 41 : HChl-CChl-CLChl
- 42 : CLChl-CChl-CLChl
- 43 : CDmso-SDmso-CDmso
- 44 : CDmso-SDmso-ODmso
- 45 : HMet-OMet-CMet
- 46 : CLCl4-CCl4-CLCl4

Regular Torsion numbers : Gromos examples of the regular torsion

Note: Gromos 43A1 only applies one torsion involving the cosine series to each pair of central atoms. The remainder of the torsions only interact via van der Waals and coulombic terms. In Towhee this means you need to set all of the other torsions to the Null torsion type of 6. I have put an X for any atom.

- 1 : X-C-C-X
- 2 : X-C-OA-X (at ring)
- 3 : X-C-OA-X (carboxyl)
- 4 : X-C-N,NT,NE,NZ,NR-X
- 5 : X-C-CR1-X (6-ring)
- 6 : X-CH1(sugar)-NR(base)-X Also used as the NULL torsion.
- 7 : O-CH1-CHn-not O
- 8 : O-CH1-CHn-O
- 9 : X-OA-P-X
- 10 : X-S-S-X
- 11 : X-OA-P-X
- 12 : X-CHn-OA(not sugar)-X
- 13 : X-CH2-S-X
- 14 : X-C,CHn,SI-NT,NL,OA(sugar)-X
- 15 : HC-C-S-X
- 16 : HC-C-C-X
- 17 : X-CHn,SI-CHn-X
- 18 : X-NR-FE-X

- 19 : X-CHn-N,NE-X
- 20 : X-CHn-C,NR(ring),CR1-X
- 21 : X-CHn-NT-X

Coulombic interactions

Gromos 43A1 uses point charges to represent the molecular electrostatic interactions. There is no automated system for assigning these charges. Instead, you need to look through the Gromos manual (van Gunsteren *et al.* 1996) and find molecules which are similar to the one you wish to study.

Improper torsions

Gromos 43A1 exclusively uses the out-of-plane version of the improper torsions. There are only three types of Gromos 43A1 improper torsions, one for enforcing tetrahedral conformations, one for enforcing planarity, and a special one for heme. In the towhee_input file this improper torsion is specified starting from the central atom, and then three atoms bonded to that atom are listed. Starting with version 2.4.5, Towhee uses the same numbering convention as Gromos 43A1. You will need to specify one of the types listed below.
Towhee Improper Torsion number : description

- 1 : planar groups
- 2 : tetrahedral centers
- 3 : heme iron

Proteins

Gromos 43A1 is designed for proteins, but I have not yet implemented this force field into the protein builder.

Return to the Towhee Capabilities web page

Send comments to: Marcus G. Martin
Last updated: September 07, 2005

MCCCS Towhee (Jaramillo *et al.* 2001)

Overview

This section covers the Jaramillo *et al.* 2001 (Jaramillo) force field as it is implemented into the towhee_ff_jaramillo file in the ForceFields directory. This force field contains parameters for Hydrofluorocarbons. Note that this is a Lennard-Jones force field that uses the 'Explicit' mixing rules. Any discrepencies (especially typos) from the published values are the sole responsibility of Marcus G. Martin, and I welcome feedback on how this implementation compares with other programs.

References for Jaramillo *et al.* 2001

This forcefield is described in a single paper.

- Jaramillo *et al.* 2001

Cui *et al.* 1998 in Towhee

The official force field name for Jaramillo *et al.* 2001 in Towhee is 'Jaramillo'. Here is the list of atom names for use in the towhee_input file, along with a brief description. These names correspond to those used in their paper. Please note that the capitalization and spacing pattern is important and must be followed exactly as listed here.

- **'C'** : carbon
- **'H'** : hydrogen
- **'F'** : flourine

Coulombic interactions

In the original paper these parameters were taken, different partial charges used for carbon atoms depending on the number of flourine atoms bonded. Therefore, for the correct assignement of the partial charges, users should consult to the paper.

Return to the Towhee Capabilities web page

Send comments to: Marcus G. Martin
Last updated: August 01, 2006

MCCCS Towhee (Last1993)

Overview

This section covers Lastoskie *et al.* parameters as they are implemented into the towhee_ff_Last1993 file in the ForceFields directory.

References for Lastoskie *et al.*

This is not a comprehensive force field. It is simply the Lennard-Jones parameters used to represent the dinitrogen (N_2) molecule in a Carbon Slit Pore.

- Lastoskie *et al.* 1993

Lastoskie *et al.* in Towhee

The official force field name for Lennard-Jonesium in Towhee is 'Last1993'. Here I list the only atom names currently in the towhee_ff_Last1993 file, along with a brief description (although the meaning should be obvious). Please note that the capitalization and spacing pattern is important and must be followed exactly as listed here.

- **'N2'** : dinitrogen represented as a single interaction site for the whole molecule.

Coulombic interactions

This molecule is uncharged.

Return to the Towhee Capabilities web page

Send comments to: Marcus G. Martin
Last updated: September 07, 2005

MCCCS Towhee (Lennard-Jonesium)

Overview

This section covers Lennard-Jonesium parameters as they are implemented into the towhee_ff_LJium file in the ForceFields directory. This force field exists to do very simple models, mostly for comparison with older simulations or theory.

Lennard-Jonesium in Towhee

The official force field name for Lennard-Jonesium in Towhee is 'LJium'. Here I list all of the Lennard-Jones atom names currently in the towhee_ff_LJium file, along with a brief description (although the meaning of the atom names should be obvious). Please note that the capitalization and spacing pattern is important and must be followed exactly as listed here.

- **'lj1.0'** : a Lennard-Jones sphere with a 1.0 diameter and an epsilon of 1 K.

Coulombic interactions

It is possible to combine the Lennard-Jones potential with point charges. Simply assign the point charges to the atoms as you see fit.

Improper torsions

There are no improper torsions set up specifically for Lennard-Jones spheres.

Proteins

There are no features to build proteins out of Lennard-Jones spheres.

Return to the Towhee Capabilities web page

Send comments to: Marcus G. Martin
Last updated: September 07, 2005

MCCCS Towhee (Lybrand-Ghosh-McCammon ions)

Overview

This section covers the Lybrand-Ghosh-McCammon (LGM) ion parameters as they are implemented into the towhee_ff_LGM file in the ForceFields directory. All of the Towhee atom types for the LGM ion force field are listed, along with a short description of their meanings. This is not really a comphrehensive stand-alone force field, but is instead a collection of parameters for some negative halide ions. Ion parameters are sometimes hard to come by in the literature so I have included these parameters for use in conjunction with other force fields (as a simulation of only negative ions is not very interesting). Note that these are Lennard-Jones (12-6) parameters and can only be combined with other Lennard-Jones (12-6) force fields. Any discrepencies (especially typos) from the published LGM force field values are the sole responsibility of Marcus G. Martin, and I welcome feedback on how this implementation compares with other programs.

References for Lybrand-Ghosh-McCammon ions

- Lybrand *et al.* 1985

Lybrand-Ghosh-McCammon in Towhee

The official force field name for Lybrand-Ghosh-McCammon in Towhee is 'LGM'. Here I list all of the LGM atom names for use in the towhee_input file, along with a brief description (although the meaning of the atom names should be obvious). Please note that the capitalization and spacing pattern is important and must be followed exactly as listed here.

- **'Br-'** : bromine ion with a -1 charge
- **'Cl-'** : chlorine ion with a -1 charge

Coulombic interactions

Atom centered point charges are used to represent the electrostatic interactions. As this is a force field for ions, the charge assignments are straight forward.

- **'Br-'** : -1.0
- **'Cl-'** : -1.0

Improper torsions

This force field only has parameters for ions so there are no improper torsions.

Proteins

This force field only has parameters for ions, and while you are welcome to use them with another protein force field, there are no amino acids in the LGM ions force field.

Return to the Towhee Capabilities web page

Send comments to: Marcus G. Martin
Last updated: September 07, 2005

MCCCS Towhee (Morrow and Maginn 2002)

Overview

This section covers the Morrow and Maginn 2002 (Morrow2002) force field as it is implemented into the towhee_ff_Morrow2002 file in the ForceFields directory. This force field was designed for an ionic liquid of 1-*n*-butyl-3-methylimidazolium [bmim] hexafluorophosphate [PF_6]. Note that this is a Lennard-Jones force field that uses the 'Lorentz-Berthelot' mixing rules. Any discrepencies (especially typos) from the published values are the sole responsibility of Marcus G. Martin, and I welcome feedback on how this implementation compares with other programs.

References for Morrow and Maginn 2002

This forcefield is described in a single paper and an erratum.

- Morrow and Maginn 2002
- Morrow and Maginn 2003 erratum

Typos and comments for Morrow and Maginn 2002

- The parameters implemented for this force field are from the erratum.
- All bonds involving hydrogen were held rigid in their work using SHAKE, and are therefore implemented here as rigid bonds.
- Full Lennard-Jones interactions are used for 1-4 interactions, while the Coulombic 1-4 terms are scaled by 1/2 [personal communication between E.J. Maginn and M.G. Martin 04-21-2006].
- The second to last angle term in Table 2 of the erratum should read N-C_7-$H_{7,8}$ instead of N-C_7-$H_{8,9}$ [personal communication between E.J. Maginn and M.G. Martin 04-21-2006].
- No value for the H-C8-C9-H dihedral was specified in their papers. It has been implemented here using the published H-C7-C8-H term.
- The authors claim that the charge on PF_6 adds up to -0.904, and the charge on bmim adds up to 0.904. However, the partial charges they list for PF_6 using assymmetric charge assignments adds up to -0.903, which does not properly balance the charge of the bmin. I have adjusted this by utlizing the stated charges on the Fluorines, and placing the remainder of the charge on the Phosphorous.

Morrow and Maginn 2002 in Towhee

The official force field name for Morrow and Maginn 2002 in Towhee is 'Morrow2002'. Here is the list of atom names for use in the towhee_input file, along with a brief description. These names correspond to those used in Morrow and Maginn 2002. Please note that the capitalization and spacing pattern is important and must be followed exactly as listed here.

- **'N1'** : Ring Nitrogen bonded to C2, C5, and C7 in bmim
- **'C2'** : Ring Carbon bonded to N1, N3, and H1 in bmim
- **'N3'** : Ring Nitrogen bonded to C2, C4, and C6 in bmim
- **'C4'** : Ring Carbon bonded to N3, C5, and H2 in bmim
- **'C5'** : Ring Carbon bonded to N1, C4, and H3 in bmim
- **'C6'** : methyl Carbon bonded to N3, H4, H5, and H6 in bmim
- **'C7'** : methylene Carbon bonded to N1, C8, H7, and H8 in bmim
- **'C8'** : methylene Carbon bonded to C7, C9, H9, and H10 in bmim
- **'C9'** : methylene Carbon bonded to C8, C10, H11, and H12 in bmim
- **'C10'** : methyl Carbon bonded to C9, H13, H14, and H15 in bmim
- **'H1'** : Hydrogen bonded to C2 in bmim

'H2' : Hydrogen bonded to C4 in bmim
- **'H3'** : Hydrogen bonded to C5 in bmim
- **'H4'** : Hydrogen bonded to C6 in bmim
- **'H5'** : Hydrogen bonded to C6 in bmim
- **'H6'** : Hydrogen bonded to C6 in bmim
- **'H7'** : Hydrogen bonded to C7 in bmim
- **'H8'** : Hydrogen bonded to C7 in bmim
- **'H9'** : Hydrogen bonded to C8 in bmim
- **'H10'** : Hydrogen bonded to C8 in bmim
- **'H11'** : Hydrogen bonded to C9 in bmim
- **'H12'** : Hydrogen bonded to C9 in bmim
- **'H13'** : Hydrogen bonded to C10 in bmim
- **'H14'** : Hydrogen bonded to C10 in bmim
- **'H15'** : Hydrogen bonded to C10 in bmim
- **'P'** : P in hexafluorophosphate
- **'F'** : generic F in hexafluorophosphate if you want to use the symmetric charge distribution version of that molecule.
- **'F1'** : specific F in hexafluorophosphate if you want to use the assymmetric charge distribution version of that molecule.
- **'F2'** : specific F in hexafluorophosphate if you want to use the assymmetric charge distribution version of that molecule.
- **'F3'** : specific F in hexafluorophosphate if you want to use the assymmetric charge distribution version of that molecule.
- **'F4'** : specific F in hexafluorophosphate if you want to use the assymmetric charge distribution version of that molecule.
- **'F5'** : specific F in hexafluorophosphate if you want to use the assymmetric charge distribution version of that molecule.
- **'F6'** : specific F in hexafluorophosphate if you want to use the assymmetric charge distribution version of that molecule.

Coulombic interactions

This force field uses point charges and has been set up to assign the point charges using the bond increment method. The charges for bmim are assigned according to Table 1 of Morrow and Maginn 2002. There are two possible charge distributions for hexafluorophosphate. If you specify each of the Fluorine atoms (F1, ..., F6) then the charges are assigned according to Table 1 of Morrow and Maginn 2002, with the exception that the P charge is 1.457 instead of the stated 1.478 (see the notes above). If you specify a generic Fluorine atom (F), then the charge on the Phosphorous is the same as in Table 1, while the Fluorines are assigned symmetric charges in order to achieve a total molecule charge of -0.904.

Return to the Towhee Capabilities web page

Send comments to: Marcus G. Martin
Last updated: April 24, 2006

MCCCS Towhee (NERD Version 1)

Overview

This section covers the NERD Version 1 United Atom (NERDv1) force field as it is implemented into the towhee_ff_NERDv1 file in the ForceFields directory. The NERD force field has had several minor changes to parameters as time as progresses so I have broken each of these changes into a different version of the force field. NERD Version 1 is mostly a subset of NERD Version 2, with the exception that the linear alkane torsional potential is slightly different between the two versions. All of the Towhee atom types for this force field are listed, along with a short description of their meanings. Note that NERDv1 is a Lennard-Jones (12-6) force field and can only be combined with other Lennard-Jones (12-6) force fields. I would like to thank Nicholas du Preez for initiating the implementation of this force field into Towhee. Any discrepencies (especially typos) from the published values are the sole responsibility of Marcus G. Martin, and I welcome feedback on how this implementation compares with other programs.

References for NERD Version 1

What I term NERD Version 1 is described in two papers.

- Nath *et al.* 1998
- Nath and de Pablo 2000

NERDv1 in Towhee

The official force field name for NERD Version 1 United Atom in Towhee is 'NERDv1'. Here I list all of the atom names for use in the towhee_input file, along with a brief description. I created these atom names in order to work with the molecule assembler. NERDv1 is a united-atom force field which lumps hydrogens onto their neighboring carbons. Note that hydrogens that are bonded to atoms other than carbon are not lumped onto their neighboring atoms. The general pattern of the names is a list of the atoms that make up the group, followed by the hybridization of the central atom, and ending in a character string that distinguishes similar atoms. Please note that the capitalization and spacing pattern is important and must be followed exactly as listed here.

- **'CH3sp3eth'** : sp^3 carbon plus the 3 hydrogens bonded to it (united-atom) in ethane.
- **'CH3sp3pro'** : sp^3 carbon plus the 3 hydrogens bonded to it (united-atom) in propane.
- **'CH3sp3isob'** : sp^3 carbon plus the 3 hydrogens bonded to it (united-atom) in isobutane.
- **'CH3sp32mb'** : sp^3 carbon plus the 3 hydrogens bonded to it (united-atom) bonded to a CH group in 2-methylbutane.
- **'CH3sp3mesc'** : sp^3 carbon plus the 3 hydrogens bonded to it (united-atom) bonded to a CH group (methyl side-chain) and not a special case listed above.
- **'CH3sp3etsc'** : sp^3 carbon plus the 3 hydrogens bonded to it (united-atom) bonded to a CH_2 group that is itself bonded to a CH group (ethyl side-chain).
- **'CH3sp3gen'** : sp^3 carbon plus the 3 hydrogens bonded to it (united-atom) and not one of the cases listed above.
- **'CH2sp3'** : sp^3 carbon plus the 2 hydrogens bonded to it (united-atom).
- **'CHsp3'** : sp^3 carbon plus the 1 hydrogen bonded to it (united-atom).
- **'CH2sp2eth'** : sp^2 carbon plus the 2 hydrogens bonded to it (united-atom) in ethene.

Coulombic interactions

There are no coulombic interactions used in NERDv1, although there are some in subsequent versions of NERD.

Improper torsions

This force field does not utilize improper torsions.

Proteins

This force field does not have all of the groups needed for protein simulations.

Return to the Towhee Capabilities web page

Send comments to: Marcus G. Martin
Last updated: September 07, 2005

MCCCS Towhee (NERD Version 2)

Overview

This section covers the NERD Version 2 United Atom (NERDv2) force field as it is implemented into the towhee_ff_NERDv2 file in the ForceFields directory. The NERD force field has had several minor changes to parameters as time as progresses so I have broken each of these changes into a different version of the force field. All of the Towhee atom types for this force field are listed, along with a short description of their meanings. Note that NERDv2 is a Lennard-Jones (12-6) force field and can only be combined with other Lennard-Jones (12-6) force fields. I would like to thank Nicholas du Preez for initiating the implementation of this force field into Towhee. Any discrepencies (especially typos) from the published values are the sole responsibility of Marcus G. Martin, and I welcome feedback on how this implementation compares with other programs.

References for NERD Version 2

What I term NERD Version 2 is fully described in three papers. Please see the NERDv1 documentation for more references to the original work for this force field.

- Nath *et al.* 2001
- Nath and Khare 2001
- Nath 2003

NERDv2 in Towhee

The official force field name for NERD Version 2 United Atom in Towhee is 'NERDv2'. Here I list all of the atom names for use in the towhee_input file, along with a brief description. I created these atom names in order to work with the molecule assembler. NERDv2 is a united-atom force field which lumps hydrogens onto their neighboring carbons. Note that hydrogens that are bonded to atoms other than carbon are not lumped onto their neighboring atoms. The general pattern of the names is a list of the atoms that make up the group, followed by the hybridization of the central atom, and ending in a character string that distinguishes similar atoms. Please note that the capitalization and spacing pattern is important and must be followed exactly as listed here.

sp^3 hybridized Carbons

- **'CH4'** : sp^3 carbon plus the 4 hydrogens bonded to it (united-atom) in methane.
- **'CH3sp3eth'** : sp^3 carbon plus the 3 hydrogens bonded to it (united-atom) in ethane.
- **'CH3sp3pro'** : sp^3 carbon plus the 3 hydrogens bonded to it (united-atom) in propane.
- **'CH3sp3isob'** : sp^3 carbon plus the 3 hydrogens bonded to it (united-atom) in isobutane (2-methylpropane).
- **'CH3sp32mb'** : sp^3 carbon plus the 3 hydrogens bonded to it (united-atom) bonded to a CH group in 2-methylbutane.
- **'CH3sp3neop'** : sp^3 carbon plus the 3 hydrogens bonded to it (united-atom) bonded to a C group in neopentane (2,2-dimethyl propane).
- **'CH3sp322db'** : sp^3 carbon plus the 3 hydrogens bonded to it (united-atom) bonded to a C group in 2,2-dimethylbutane.
- **'CH3sp3pene'** : sp^3 carbon plus the 3 hydrogens bonded to it (united-atom) bonded to an sp^2 CH group in propene.
- **'CH3sp3mesc'** : sp^3 carbon plus the 3 hydrogens bonded to it (united-atom) bonded to a CH group (methyl side-chain) and not a special case listed above.

- **'CH3sp3etsc'** : sp^3 carbon plus the 3 hydrogens bonded to it (united-atom) bonded to a CH_2 group that is itself bonded to a CH group (ethyl side-chain).
- **'CH3sp3gen'** : sp^3 carbon plus the 3 hydrogens bonded to it (united-atom) and not one of the cases listed above.
- **'CH2sp3'** : sp^3 carbon plus the 2 hydrogens bonded to it (united-atom).
- **'CHsp3'** : sp^3 carbon plus the 1 hydrogen bonded to it (united-atom).
- **'Csp3'** : sp^3 carbon that is not bonded to any hydrogens.

sp^2 hybridized Carbons

- **'CH2sp2eth'** : sp^2 carbon plus the 2 hydrogens bonded to it (united-atom) in ethene.
- **'CH2sp2prim'** : sp^2 carbon plus the 2 hydrogens bonded to it (united-atom) in a primary alkene.
- **'CHsp2'** : sp^2 carbon plus the 1 hydrogen bonded to it (united-atom) in a primary alkene.

Hydrogen

- **'H_s'** : hydrogen bonded to sulfur.

Sulfur

- **'S_sh2'** : sulfur in dihydrogen sulfide.

Coulombic interactions

Most atoms do not have a charge when using the NERDv2 forcefield. Exceptions are listed below.

- **'H_s'** : 0.124
- **'S_sh2'** : -0.248

Improper torsions

This force field does not utilize improper torsions.

Proteins

This force field does not have all of the groups needed for protein simulations.

Return to the Towhee Capabilities web page

Send comments to: Marcus G. Martin
Last updated: September 07, 2005

MCCCS Towhee (NERD Version 3)

Overview

This section covers the NERD Version 3 United Atom (NERDv3) force field as it is implemented into the towhee_ff_NERDv3 file in the ForceFields directory. The NERD force field has had several minor changes to parameters as time as progresses so I have broken each of these changes into a different version of the force field. All of the Towhee atom types for this force field are listed, along with a short description of their meanings. Note that NERDv3 is a Lennard-Jones (12-6) force field and can only be combined with other Lennard-Jones (12-6) force fields. I would like to thank Nicholas du Preez for initiating the implementation of this force field into Towhee. Any discrepencies (especially typos) from the published values are the sole responsibility of Marcus G. Martin, and I welcome feedback on how this implementation compares with other programs.

References for NERD Version 3

What I term NERD Version 3 is described in a single paper (listed here) plus all of the original NERDv1 and NERDv2 references. Note that some of the terms in those original works have been superceded by later publications.

- Khare *et al.* 2004

NERDv3 in Towhee

The official force field name for NERD Version 2 United Atom in Towhee is 'NERDv3'. Here I list all of the atom names for use in the towhee_input file, along with a brief description. I created these atom names in order to work with the molecule assembler. NERDv3 is a united-atom force field which lumps hydrogens onto their neighboring carbons. Note that hydrogens that are bonded to atoms other than carbon are not lumped onto their neighboring atoms. The general pattern of the names is a list of the atoms that make up the group, followed by the hybridization of the central atom, and ending in a character string that distinguishes similar atoms. Please note that the capitalization and spacing pattern is important and must be followed exactly as listed here.

sp^3 hybridized Carbons

- **'CH4'** : sp^3 carbon plus the 4 hydrogens bonded to it (united-atom) in methane.
- **'CH3sp3eth'** : sp^3 carbon plus the 3 hydrogens bonded to it (united-atom) in ethane.
- **'CH3sp3pro'** : sp^3 carbon plus the 3 hydrogens bonded to it (united-atom) in propane.
- **'CH3sp3isob'** : sp^3 carbon plus the 3 hydrogens bonded to it (united-atom) in isobutane (2-methylpropane).
- **'CH3sp32mb'** : sp^3 carbon plus the 3 hydrogens bonded to it (united-atom) bonded to a CH group in 2-methylbutane.
- **'CH3sp3neop'** : sp^3 carbon plus the 3 hydrogens bonded to it (united-atom) bonded to a C group in neopentane (2,2-dimethyl propane).
- **'CH3sp322db'** : sp^3 carbon plus the 3 hydrogens bonded to it (united-atom) bonded to a C group in 2,2-dimethylbutane.
- **'CH3sp3pene'** : sp^3 carbon plus the 3 hydrogens bonded to it (united-atom) bonded to an sp^2 CH group in propene.
- **'CH3sp3mesc'** : sp^3 carbon plus the 3 hydrogens bonded to it (united-atom) bonded to a CH group (methyl side-chain) and not a special case listed above.
- **'CH3sp3etsc'** : sp^3 carbon plus the 3 hydrogens bonded to it (united-atom) bonded to a

CH_2 group that is itself bonded to a CH group (ethyl side-chain).

- **'CH3sp3gen'** : sp^3 carbon plus the 3 hydrogens bonded to it (united-atom) and not one of the cases listed above.
- **'CH2sp3'** : sp^3 carbon plus the 2 hydrogens bonded to it (united-atom).
- **'CHsp3'** : sp^3 carbon plus the 1 hydrogen bonded to it (united-atom).
- **'Csp3'** : sp^3 carbon that is not bonded to any hydrogens.

sp^2 hybridized Carbons

- **'CH2sp2eth'** : sp^2 carbon plus the 2 hydrogens bonded to it (united-atom) in ethene.
- **'CH2sp2prim'** : sp^2 carbon plus the 2 hydrogens bonded to it (united-atom) in a primary alkene.
- **'CHsp2'** : sp^2 carbon plus the 1 hydrogen bonded to it (united-atom) in a primary alkene.

Hydrogen

- **'H_o'** : hydrogen bonded to oxygen.
- **'H_s'** : hydrogen bonded to sulfur.

Oxygen

- **'Osp3'** : sp^3 hybridized oxygen single bonded to two atoms.

Sulfur

- **'S_sh2'** : sulfur in dihydrogen sulfide.

Coulombic interactions

Most atoms do not have a charge when using the NERDv3 forcefield. Exceptions are listed below.

SH^2 (from Nath 2003)

- **'H_s'** : 0.124
- **'S_sh2'** : -0.248

alcohols (from Khare *et al.* 2004)

- **'H_o'** : 0.420
- **'Osp3'** : -0.710
- **'CH2sp3'** (bonded to the oxygen): 0.290

Improper torsions

This force field does not utilize improper torsions.

Proteins

This force field does not have all of the groups needed for protein simulations.

Return to the Towhee Capabilities web page

Send comments to: Marcus G. Martin
Last updated: September 07, 2005

MCCCS Towhee (OPLS-aa)

Overview

This section covers the OPLS-aa force field as it is implemented into the towhee_ff_OPLS-aa file in the ForceFields directory. All of the Towhee atom types for the OPLS-aa force field are listed, along with a short description of their meanings. For more information about the OPLS-aa force field see the Jorgensen group home page. Note that OPLS-aa is a Lennard-Jones (12-6) force field and can only be combined with other Lennard-Jones (12-6) force fields. The rather famous OPLS water models are not included here as they are provided in their own force field files. Please see the TIP3P, TIP4P, and TIP5P web pages for more information about those water models. Note that OPLS-aa was preceded by a united-atom version of the force field called OPLS-ua and has been reparameterized into what I call OPLS-2001. I would like to acknowledge W.L. Jorgensen for kindly providing me with an electronic copy of the OPLS-aa parameters. Any discrepencies (especially typos) from the published OPLS-aa force field values are the sole responsibility of Marcus G. Martin, and I welcome feedback on how this implementation compares with other programs.

References for OPLS-aa

OPLS-aa is published in a series of papers

- Chandrasekhar *et al.* 1984
- Pranata *et al.* 1991
- Kaminski *et al.* 1994
- Jorgensen *et al.* 1996
- Jorgensen *personal communication* 1996
- Damm *et al.* 1997
- Jorgensen and McDonald 1998
- McDonald and Jorgensen 1998
- McDonald *et al* 1998
- Rizzo and Jorgensen 1999
- Jorgensen *personal communication* 2001

OPLS-aa in Towhee

The official force field name for OPLS-aa in Towhee is 'OPLS-aa'. Here I list all of the OPLS-aa atom names for use in the towhee_input file, along with a brief description taken from the OPLS-aa literature. Notice that OPLS-aa uses almost the same naming conventions as the Amber param96 force field. This is not a coincedence as these force fields utilize many of the same bonded interaction terms. I have modified some of the atom names from the standard OPLS-aa in order to create unique type names. This was needed because OPLS-aa occasionally uses different bonded interactions for atoms which have the same atom name for nonbonded interactions. I do not always distinguish between the original OPLS-aa comments, and my modifications as I have created several new atom names. Some of my comments are placed in [square brackets]. Please note that the capitalization and spacing pattern is important and must be followed exactly as listed here.

Bromine

- **'Br-'** : bromine ion (charge -1)[uses the parameters from the LGM force field]

Carbon

- **'C a'** : carbon in O-C=O acid
- **'C k'** : carbon in C=O not bonded to N, and not an acid group
- **'C n'** : carbon in N-C=O

'C*' : aromatic C in 5-membered ring next to two carbons

- **'CA~'** : neutral aromatic carbon not at the junction of 5-membered and 6-membered rings
- **'CA+'** : aromatic carbon in guanidinium C+
- **'CAj'** : neutral aromatic carbon at the juction of 5-membered and 6-membered rings
- **'CAxna'** : special parameters for nucleotide bases. aromatic C in DAP C2,C3,C4 pyridine; Cytosine C4,C5,C6
- **'CBxna'** : special parameters for nucleotide bases. Adenine C4,C5,C6; Guanine C2,C4,C5
- **'CB'** : aromatic C at juntion of 5-membered and 6-membered rings
- **'CKxna'** : special parameters for nucleotide bases. Adenine C8 Guanine
- **'CM'** : alkene sp^2 carbon (non-aromatic)
- **'CMxna'** : special parameters for nucleotide bases. Carbon in Uracil C5,C6
- **'CN'** : aromatic carbon in tryptophan at position CE2.
- **'CO'** : sp^3 anomeric carbon (bonded to ether O and alcohol O). this type shows up in carbohydrates
- **'CQ'** : sp^{2}C in 6-membered ring between deprotonated N's
- **'CR'** : aromatic C in 5-membered ring next to two nitrogens
- **'CS'** : aromatic C which does not have 120 degree bond angles with all of its neighbors. Examples are C3 in pyrrole and C3 in furan
- **'CT'** : aliphatic sp^3 hybrid carbon
- **'CTf'** : aliphatic sp^3 hybrid carbon bonded to F
- **'CTxna'** : special parameters for nucleotide bases. Thymine C-C5, 9-Me A or G C-N9
- **'CU'** : aromatic C which abuts NB in a heteroatom-heteroatom bond. Examples are C3 in pyrazole and C3 in isoxazole
- **'CV'** : aromatic C in 5-membered ring next to C and deprotonated N
- **'CW'** : sp^2 aromatic C in 5-membered ring next to C and NH

Chlorine

- **'Cl'** : chlorine bonded to a carbon
- **'Cl-'** : chlorine ion (charge -1)

Fluorine

- **'F'** : fluorine nonionic
- **'F-'** : fluorine ion (charge -1)

Hydrogen

- **'H'** : H attached to N
- **'H2xna'** : special parameters for nucleotide bases. DAP H-amine, Cytosine, Adenine
- **'HA'** : H bonded to an aromatic ring
- **'HC~1'** : hydrogen attached to an sp^3 carbon in most cases, see special cases of hydrogen below
- **'HC~2'** : H bonded to a non-aromatic sp^2 carbon which is not bonded to a N, not double bonded to O, and is not bonded to C=O
- **'HC~3'** : H bonded to a C=O or bonded to a C which is bonded to a C=O
- **'HC~4'** : H bonded to a CT that is bonded to NT
- **'HCxna'** : special parameters for nucleotide bases. DAP H3,H4; Uracil H-C5,H-C6; Thymine H-CC5; Cytosine H-C5,H-C6; Adenine H-C2,H-C8; Guanine
- **'HO'** : H attached to O in alcohol
- **'HS'** : H attached to S

Iodine

- **'I-'** : iodine ion (charge -1)

Lithium

- **'Li+'** : lithium ion (charge +1)

Nitrogen

- **'N'** : N in an amide
- **'N2'** : sp^2 N of aromatic amines and guanidinium ions
- **'N3~'** : sp^3 neutral N of amines. NOTE: the N3 parameters are obsoleted by the NT parameters. I have included them here in case you want to study these older parameters, but I would recomend using the NT versions.
- **'N3+'** : sp^3 N ammonium ions (charge +1). NOTE: the N3 parameters are obsoleted by NT parameters. I have included them here in case you want to study these older parameters, but I would recomend using the NT versions.
- **'NA'** : sp^2 aromatic N with H attached
- **'NB'** : sp^2 N in 5-membered ring, deprotonated
- **'NC'** : sp^2 N in 6-membered ring, deprotonated
- **'NO'** : sp^2 N in a nitro group
- **'NT~1'** : sp^3 N in ammonia
- **'NT+1'** : sp^3 N in ammonium ion (charge +1)
- **'NT~2'** : sp^3 N in primary, secondary, tertiary amine
- **'NT+2'** : sp^3 N in primary, secondary, tertiary ammonium ion (charge +1)

Oxygen

- **'O'** : oxygen in C=O, not an acidic site
- **'O2'** : oxygen double bonded to carbon in COO- or COOH
- **'OHa'** : oxygen bonded to H in RCOOH
- **'OHm'** : oxygen bonded to H in a mono-alcohol
- **'OHp'** : oxygen bonded to H in polyols or phenol
- **'OS'** : sp^3 O in an ether or acetal

Sodium

- **'Na+'** : sodium ion (charge +1)

Sulfur

- **'S'** : S in sulfide, disulfide
- **'SH'** : S in thiols

Coulombic interactions

OPLS-aa uses atom-centered point charges to represent the electrostatic interactions. I do not know of an automated way to assign these point charges. Instead, you need to look through the OPLS-aa literature to find molecules with similar moieties to the ones on the molecule you wish to simulate.

Improper torsions

OPLS-aa uses stereocenter improper torsions to enforce planarity in aromatic rings, around amide nitrogens, and around other sp2 carbons. Remember that you can set the improper type to 0 to have the code automatically determine the improper type (so long as inpstyle is 2). Improper torsions for proteins are exactly the same as implemented in AMBER96 - even though OPLS puts the stereocenter in the second position of the torsion, you still define them in towhee putting it in the first position (towhee automatically adjusts the parameter order for you).

Proteins

The OPLS-aa protein builder has been implemented with all charges verbatim from WLJ 1996. (Note that the sidechain charges on Met and Cys were later revised in OPLS-2001). Because many of the bonded parameters are unpublished, they have been cobbled together from other OPLS-aa implementations, namely ffoplsaabon.itp in GROMACS 3.2.1 and oplsaa.prm in

TINKER 4.0- the origin of added parameters is referenced in the comments of ffoplsaa.F. Most of the missing parameters merely required additional aliases to existing entries because of the above naming convention. Note that GROMACS uses nanometers and KJoules instead of Angstroms and Kcals so unit conversions are necessary when comparing Towhee and GROMACS results. Here is a complete list of the codes for the 20 amino acids, plus some other functional groups that work with the protein builder.

- 'a0' alanine
- 'c0' cysteine with hydrogen on the sulfur
- 'cs' cysteine in a disulfide bond
- 'd-' aspartic acid deprotonated
- 'e-' glutamic acid deprotonated
- 'f0' phenylalanine
- 'g0' glycine
- 'h+' histidine both N protonated
- 'hd' histidine neutral with only N_d protonated
- 'he' histidine neutral with only N_e protonated
- 'i0' isoleucine
- 'k+' lysine protonated
- 'l0' leucine
- 'm0' methionine
- 'n0' asparagine
- 'p0' proline
- 'q0' glutamine
- 'r+' arginine protonated
- 's0' serine
- 't0' threonine
- 'v0' valine
- 'w0' tryptophan. Despite a great deal of effort this amino acid is still not working properly with this forcefield as there are many parameters that are missing. You will get an error if you include this amino acid for this forcefield.
- 'y0' tyrosine
- 'za' acetyl cap on the N-terminus
- 'zm' amide cap (NH_2) on the C-terminus.
- 'zn' N-methylamide cap on the C-terminus
- 'zf' formaldahyde cap on the N-terminus

Return to the Towhee Capabilities web page

Send comments to: Marcus G. Martin
Last updated: June 28, 2006

MCCCS Towhee (OPLS-ua)

Overview

This section covers the OPLS-ua force field as it is implemented into the towhee_ff_OPLS-ua file in the ForceFields directory. All of the Towhee atom types for the OPLS-ua force field are listed, along with a short description of their meanings. For more information about the OPLS family of force fields see the Jorgensen group home page. Note that OPLS-ua is a Lennard-Jones (12-6) force field and can only be combined with other Lennard-Jones (12-6) force fields. The rather famous OPLS water models are not included here as they are provided in their own force field files. Please see the TIP3P, TIP4P, and TIP5P web pages for more information about those water models. Note that OPLS-ua was followed up with an all-atom version of the force field called OPLS-aa and has been reparameterized into what I call OPLS-2001. Any discrepencies (especially typos) from the published OPLS-ua force field values are the sole responsibility of Marcus G. Martin, and I welcome feedback on how this implementation compares with other programs.

References for OPLS-ua

OPLS-ua is published in a series of papers

- Chandrasekhar *et al.* 1984
- Cournoyer and Jorgensen 1984
- Jorgensen *et al.* 1984
- Jorgensen and Swenson 1985
- Jorgensen 1986 alcohol
- Jorgensen 1986 sulfur
- Jorgensen and Briggs 1988
- Jorgensen *et al.* 1990

OPLS-ua in Towhee

The official force field name for OPLS-ua in Towhee is 'OPLS-ua'. Here I list all of the OPLS-ua atom names for use in the towhee_input file, along with a brief description taken from the OPLS-ua literature. I have used my own judgement in creating naming conventions as they are not well defined in the literature. Please note that the capitalization and spacing pattern is important and must be followed exactly as listed here.

Carbon (CH_3 United-Atoms)

- **'CH4'** : United-atom Methane
- **'CH3-C1'** : CH_3 united-atom consisting of 1 C and 3 H that is bonded to another carbon that is not in turn bonded to any other carbons(example: ethane methyl group, acetonitrile methyl group)
- **'CH3-C2'** : CH_3 united-atom consisting of 1 C and 3 H that is bonded to another carbon that has 2 non-H neighbors (example: n-alkane methyl group), or is bonded to a sulfur that is in turn bonded to a hydrogen (example: thiols).
- **'CH3-C3'** : CH_3 united-atom consisting of 1 C and 3 H that is bonded to another carbon that has 3 non-H neighbors (example: isobutane methyl group).
- **'CH3-C4'** : CH_3 united-atom consisting of 1 C and 3 H that is bonded to another carbon that has 4 non-H neighbors (example: neopentane methyl group)
- **'CH3-X'** : CH_3 united-atom consisting of 1 C and 3 H that is bonded to an atom other than carbon that is in turn bonded to another non-hydrogen atom or more than one hydrogram atom (example: sulfides, disulfides, N-methylacetamide, dimethylformamide,

methyl-acetate).
Carbon (CH_2 United-Atoms)

- **'CH2sp3-SR'** : CH_2 united-atom consisting of 1 C and 2 H with an *sp*^{3} hybridization on the C that is bonded to a sulfur that is in turn bonded to another non-hydrogen atom (example: sulfides and disulfides).
- **'CH2sp3-N'** : CH_2 united-atom consisting of 1 C and 2 H with an *sp*^{3} hybridization on the C that is bonded to a nitrogen (example: C_{alpha} in glycine, C in proline).
- **'CH2sp3'** : CH_2 united-atom consisting of 1 C and 2 H with an *sp*^{3} hybridization on the C.
- **'CH2sp2'** : CH_2 united-atom consisting of 1 C and 2 H with an *sp*^{2} hybridization on the C.

Carbon (CH United-Atoms)

- **'CHsp3-N'** : CH united-atom consisting of 1 C and 1 H with an *sp*^{3} hybridization on the C that is bonded to a Nitrogen (example: C_{alpha} in Ala, C_{alpha} in Pro).
- **'CHsp3'** : CH united-atom consisting of 1 C and 1 H with an *sp*^{3} hybridization on the C.
- **'CHsp2-N'** : CH united-atom consisting of 1 C and 1 H with an *sp*^{2} hybridization on the C that is bonded to a Nitrogen (example: formamide).
- **'CHsp2'** : CH united-atom consisting of 1 C and 1 H with an *sp*^{2} hybridization on the C.
- **'CHarom'** : CH united-atom consisting of 1 C and 1 H with an aromatic C.

Carbon Atoms

- **'Csp3'** : C atom consisting of 1 C bonded to 0 H with an *sp*^{3} hybridization on the C.
- **'Csp2'** : C atom consisting of 1 C bonded to 0 H with an *sp*^{2} hybridization on the C.
- **'Csp'** : C atom consisting of 1 C bonded to 0 H with an *sp* hybridization on the C (example: central C in acetonitrile).

Chlorine

- **'Cl-'** : Cl ion with a negative 1.0 charge.

Fluorine

- **'F'** : F in Hydrogen Fluoride.
- **'F-'** : F ion with a negative 1.0 charge.

Hydrogen

- **'H'** : H atom bonded to something other than a carbon.

Lithium

- **'Li+'** : Li ion with a positive 1.0 charge.

Nitrogen

- **'Nsp2'** : Nitrogen bonded to 3 other atoms in a trigonal planar arrangement (example: amides and amino acids).
- **'Ns'** : Nitrogen triple bonded to one other atom (example: acetonitrile).

Oxygen

- **'Osp3'** : sp^3 hybridization on an O atom that is bonded to two other atoms.

Sodium

- **'Na+'** : Na ion with a positive 1.0 charge.

Sulfur

- **'S-HSH'** : sulfur bonded to 2 hydrogen atoms.
- **'S-RSR'** : sulfur bonded to 2 atoms where at least one of those atoms is not a hydrogen.

Coulombic interactions

OPLS-ua uses atom-centered point charges to represent the electrostatic interactions. You need to look through the OPLS-ua literature to find molecules with similar moieties to the ones on the molecule you wish to simulate. Here I list some examples.

Alcohols (from Jorgensen 1986 alcohols)

- **'H'**: -0.700
- **'Osp3'**: 0.435
- **'C'** that is bonded directly to the oxygen: 0.265

Thiols (from Jorgensen 1986 sulfur)

- **'H'**: 0.27
- **'S-RSR'**: -0.45
- **'CH*'** bonded to the thiol sulfur: 0.18

Sulfides (from Jorgensen 1986 sulfur)

- **'CH3-SR'** or **'CH2sp3-SR'**: 0.235
- **'S-RSR'**: -0.47
- **'CH3-SR'** or **'CH2sp3-SR'**: 0.235

Disulfides (from Jorgensen 1986 sulfur)

- **'CH3-SR'** or **'CH2sp3-SR'**: 0.30
- **'S-RSR'**: -0.30
- **'S-RSR'**: -0.30
- **'CH3-SR'** or **'CH2sp3-SR'**: 0.30

SH_2 (from Jorgensen 1986 sulfur)

- **'H'**: 0.235
- **'S-HSH'**: -0.47
- **'H'**: 0.235

HF (from Cournoyer and Jorgensen 1984)

- **'H'**: 0.64942
- **'F'**: -0.64942

Parameters for Amides and Peptides (from Jorgensen and Swenson 1985)

- **'Nsp2'** (1^o) formamide, Asn: -0.85
- **'Nsp2'** (2^o) NMA, Ala: -0.57
- **'Nsp2'** (3^o) DMF, Pro: -0.57
- **'H'** ($N1^o$) formamide, Asn: 0.425
- **'H'** ($N2^o$) NMA, Ala: 0.37
- **'Osp2'** (C=O) amides: -0.50
- **'Csp2'** (C=O) NMA, Ala: 0.50
- **'CHsp2-N'** (CH=O) formamide: 0.50
- **'CH3-N'** ($N2^o$) NMA: 0.20
- **'CH3-N'** ($N3^o$) DMF: 0.285
- **'CH2sp3-N'** C_{alpha} in Gly: 0.20
- **'CHsp3-N'** C_{alpha} in Ala: 0.20
- **'CHsp3-N'** C_{alpha} in Pro: 0.285
- **'CH2sp3-N'** Pro: 0.285
- **'CH3-N'** (C=O) NMA: 0.0

Acetonitrile (from Jorgensen and Briggs 1988)

- **'CH3-C1'**: 0.15
- **'Csp'**: 0.28
- **'Ns'**: -0.43

Improper torsions

There are no improper torsions currently implemented for this force field.

Proteins

The protein builder is not implemented for this force field.

Return to the Towhee Capabilities web page

Send comments to: Marcus G. Martin
Last updated: September 07, 2005

MCCCS Towhee (OPLS-1996)

Overview

This section covers the OPLS-1996 force field as it is implemented into the towhee_ff_OPLS-1996 file in the ForceFields directory. All of the Towhee atom types for this force field are listed, along with a short description of their meanings. For more information about the OPLS family of force fields see the Jorgensen group home page. Note this is a Lennard-Jones force field and can only be combined with other Lennard-Jones force fields. The rather famous OPLS water models are not included here as they are provided in their own force field files. Please see the TIP3P, TIP4P, and TIP5P web pages for more information about those water models. Note that the all-atom OPLS force fields were preceded by a united-atom version of the force field called OPLS-ua. There are several similar, but slightly different, versions of the OPLS all-atom forcefields implemented into Towhee. See OPLS-aa and OPLS-2001 for descriptions of these other versions. I would like to acknowledge W.L. Jorgensen for kindly providing me with an electronic copy of the OPLS-1996 parameters. Any discrepencies (especially typos) from the published OPLS-1996 force field values are the sole responsibility of Marcus G. Martin, and I welcome feedback on how this implementation compares with other programs.

References for OPLS-1996

This implementation is taken from an electronic version of the OPLS parameters that is dated 12/96. That file references the following publication.

- Jorgensen *et al.* 1996
- Jorgensen *personal communication* 1996

OPLS-1996 in Towhee

The official force field name for OPLS-1996 in Towhee is 'OPLS-1996'. Here I list all of the atom names for use in the towhee_input file, along with a brief description taken from the electronic file. This file contains a mixture of united-atom and all-atom parameters. The Towhee name is a combination of the two letter descriptor and the integer count for each atom type provided in the electronic file. The capitalization and spacing pattern is important and must be followed exactly as listed here.

Types 1-134 are from the united-atom force field. Some are needed for the UA solvents.
Note: for UA amide parameters

- NMA - types 1,2,3,4,7,39
- formamide 131,2,12,13
- DMF 131,2,3,132
- acetamide 1,2,7,12,13

- **'C -1'** :
- **'O -2'** :
- **'N -3'** :
- **'H -4'** :
- **'C2-5'** :
- **'CH-6'** :
- **'C3-7'** :
- **'CH-8'** :
- **'C2-9'** :
- **'C3-10'** :
- **'CD-11'** :

'N -12' :
- **'H -13'** :
- **'CH-14'** :
- **'C2-15'** :
- **'C2-16'** :
- **'C -17'** :
- **'O2-18'** :
- **'C2-19'** :
- **'N3-20'** :
- **'H3-21'** :
- **'C2-22'** :
- **'OH-23'** :
- **'HO-24'** :
- **'CH-25'** :
- **'C -26'** :
- **'C2-27'** :
- **'C2-28'** :
- **'CH-29'** :
- **'CH-30'** :
- **'C2-31'** :
- **'SH-32'** :
- **'HS-33'** :
- **'C2-34'** :
- **'S -35'** :
- **'C3-36'** :
- **'C2-37'** :
- **'S -38'** :
- **'C3-39'** :
- **'NA-40'** :
- **'H -41'** :
- **'NB-42'** :
- **'CP-43'** :
- **'CF-44'** :
- **'CC-45'** :
- **'NA-46'** :
- **'H -47'** :
- **'CP-48'** :
- **'CG-49'** :
- **'CB-50'** :
- **'N2-51'** :
- **'H3-52'** :
- **'CA-53'** :
- **'N2-54'** :
- **'H3-55'** :
- **'C2-56'** :
- **'C2-57'** :
- **'C -58'** : C in COOR ester JPC3315(91)
- **'O -59'** : O= in COOR ester
- **'CH-60'** :
- **'C2-61'** :
- **'OS-62'** : O- in COOR ester
- **'C3-63'** : CH3 in COOCH3
- **'CT-64'** :
- **'C3-65'** :

Types 66-77 JACS,106,6638 (1984)

- **'C4-66'** : CH4
- **'C3-67'** : CH3 (C1) ETHANE
- **'C3-68'** : CH3 (C2) N-ALKANES
- **'C3-69'** : CH3 (C3) ISOBUTANE
- **'C3-70'** : CH3 (C4) NEOPENTANE
- **'C2-71'** : CH2 (SP3) ALKANES
- **'C9-72'** : CH2 (SP2) 1-ALKENES
- **'CH-73'** : CH (SP3) ISOBUTANE
- **'C8-74'** : CH (SP2) 2-ALKENES
- **'CD-75'** : CH (AROM) BENZENOID united atom
- **'CT-76'** : C (SP3) NEOPENTANE
- **'C7-77'** : C (SP2) ISOBUTENE
- **'OH-78'** : O ALCOHOLS JPC,90,1276 (1986)
- **'HO-79'** : H(O) ALCOHOLS JPC,90,1276 (1986)
- **'C3-80'** : CH3 IN METHANOL JPC,90,1276 (1986)
- **'C2-81'** : CH2 IN ETHANOL JPC,90,1276 (1986)
- **'SH-82'** : S IN H2S JPC,90,6379 (1986)
- **'SH-83'** : S IN RSH JPC,90,6379 (1986)
- **'S -84'** : S IN RSR JPC,90,6379 (1986)
- **'S -85'** : S IN RSSR JPC,90,6379 (1986)
- **'HS-86'** : H IN H2S JPC,90,6379 (1986)
- **'HS-87'** : H(S) IN RSH JPC,90,6379 (1986)
- **'C3-88'** : CH3 IN CH2SH JPC,90,6379 (1986)
- **'C2-89'** : CH2 IN CH3CHSSH JPC,90,6379 (1986)
- **'C3-90'** : CH3 IN CH3SR JPC,90,6379 (1986)
- **'C2-91'** : CH2 IN RCH2SR JPC,90,6379 (1986)
- **'C3-92'** : CH3 IN CH3SSR JPC,90,6379 (1986)
- **'C2-93'** : CH2 IN RCH2SSR JPC,90,6379 (1986)
- **'Ar-97'** : Ar - Verlet & Weis, Mol Phys. 24,1013 (1972)
- **'Kr-98'** : Kr - Verlet & Weis, Mol Phys. 24,1013 (1972)
- **'Xe-99'** : Xe - Verlet & Weis, Mol Phys. 24,1013 (1972)
- **'N3-101'** : N (NH4+) JPC,90,2174 (1986)
- **'N3-102'** : N (RNH3+) JPC,90,2174 (1986)
- **'H3-104'** : H (NH4+) JPC,90,2174 (1986)
- **'H3-105'** : H (RNH3+) JPC,90,2174 (1986)
- **'C3-106'** : CH3 (CH3NH3+) JPC,90,2174 (1986) United-atom
- **'C3-107'** : CH3 ((CH3)4N+) JPC,90,2174 (1986) United-atom
- **'OS-108'** : ETHER O JCC,11,958 (1990)
- **'C3-109'** : ETHER CH3 (-O) JCC,11,958 (1990)
- **'C2-110'** : ETHER CH2 (-O) JCC,11,958 (1990)
- **'C2-118'** : CH2 Methylenechloride C-CL = 1.772
- **'Cl-119'** : Cl Methylenechloride Cl-C-Cl = 111.8
- **'CH-120'** : CH Chloroform JPC,94,1683 (1990)
- **'Cl-121'** : Cl Chloroform C-Cl = 1.758 Cl-C-Cl = 111.3
- **'CT-122'** : C CCl4
- **'Cl-123'** : Cl CCl4
- **'Ne-129'** : Neon - Hirschfelder (Wiley,1954)
- **'He-130'** : Helium - Hirschfelder (Wiley,1954)
- **'C -131'** : C in C=O for UA formamide, DMF
- **'C3-132'** : CH3 in HCON(CH3)2 DMF

All-atom parameters start at 135.

- **'CT-135'** : CH3 all-atom C: alkanes
- **'CT-136'** : CH2 all-atom C: alkanes

- **'CT-137'** : CH all-atom C: alkanes
- **'CT-138'** : CH4 all-atom C: methane
- **'CT-139'** : C all-atom C: alkanes
- **'HC-140'** : H all-atom H: alkanes
- **'CM-141'** : alkene C (R2-C=) all atom
- **'CM-142'** : alkene C (RH-C=) all atom
- **'CM-143'** : alkene C (H2-C=) all atom
- **'HC-144'** : alkene H (H-C=) all atom
- **'CA-145'** : Benzene C - 12 site JACS, 112, 4768-90
- **'HA-146'** : Benzene H - 12 site
- **'CA-147'** : Naphthalene fusion C (C9)
- **'CT-148'** : all-atom C: CH3, toluene
- **'CT-149'** : all -atom C: CH2, ethyl benzene
- **'OH-154'** : all-atom O: mono alcohols
- **'HO-155'** : all-atom H(O): mono alcohols
- **'HC-156'** : all-atom H(C): methanol
- **'CT-157'** : all-atom C: CH3 & CH2, alcohols
- **'CT-158'** : all-atom C: CH, alcohols
- **'CT-159'** : all-atom C: C, alcohols
- **'CT-160'** : CH2 Trifluoroethanol
- **'CT-161'** : CF3 Trifluoroethanol
- **'OH-162'** : OH Trifluoroethanol
- **'HO-163'** : HO Trifluoroethanol
- **'F -164'** : F Trifluoroethanol
- **'HC-165'** : H Trifluoroethanol
- **'CA-166'** : C(OH) phenol Use with all atom C, H 145 & 146
- **'OH-167'** : O phenol Use with all atom C, H 145 & 146
- **'HO-168'** : H phenol Use with all atom C, H 145 & 146
- **'OH-169'** : O: diols
- **'HO-170'** : H(O) diols
- **'OS-180'** : O: ether
- **'CT-181'** : C(H3OR): methyl ether
- **'CT-182'** : C(H2OR): ethyl ether
- **'CT-183'** : C(HOR): i-Pr ether
- **'CT-184'** : C(OR): t-Bu ether
- **'HC-185'** : H(COR): alpha H ether
- **'OS-186'** : O: acetal
- **'OH-187'** : O(H): hemiacetal
- **'HO-188'** : H(O): hemiacetal
- **'CO-189'** : C(H2O2): acetal OCH2O
- **'HC-190'** : H(CHO2): acetal OCH2O
- **'CO-191'** : C(H2O2): hemiacetal OCH2OH
- **'HC-192'** : H(CHO2): hemiacetal OCH2OH
- **'CO-193'** : C(HCO2): acetal OCHRO
- **'HC-194'** : H(CHO2): acetal OCHRO
- **'CO-195'** : C(HCO2): hemiacetal OCHROH
- **'HC-196'** : H(C2O2): hemiacetal OCHROH
- **'CO-197'** : C(C2O2): acetal OCRRO
- **'CO-198'** : C(C2O2): hemiacetal OCRROH
- **'SH-200'** : all-atom S: thiols
- **'SH-201'** : S IN H2S JPC,90,6379 (1986)
- **'S -202'** : all-atom S: sulfides
- **'S -203'** : all-atom S: disulfides
- **'HS-204'** : all-atom H(S): thiols

- **'HS-205'** : H IN H2S JPC,90,6379 (1986)
- **'CT-206'** : all-atom C: CH2, thiols
- **'CT-207'** : all-atom C: CH, thiols
- **'CT-208'** : all-atom C: C, thiols
- **'CT-209'** : all-atom C: CH3, sulfides
- **'CT-210'** : all-atom C: CH2, sulfides
- **'CT-211'** : all-atom C: CH, sulfides
- **'CT-212'** : all-atom C: C, sulfides
- **'CT-213'** : all-atom C: CH3, disulfides
- **'CT-214'** : all-atom C: CH2, disulfides
- **'CT-215'** : all-atom C: CH, disulfides
- **'CT-216'** : all-atom C: C, disulfides
- **'NT-220'** : N in RNH2 amine JPC,94,1683 (1990)
- **'H2-221'** : H in RNH2 amine
- **'CT-222'** : C in CH3NH2; H on C is type 140
- **'CT-223'** : C in RCH2NH2 and Gly CA
- **'CT-224'** : C in R2CHNH2 and Ala CA
- **'CT-225'** : C in R3CNH2 and Aib CA
- **'CA-226'** : C(NH2) aniline
- **'NT-227'** : N aniline
- **'H2-228'** : H(N) aniline
- **'NT-229'** : N in R3N trimethylamine
- **'CT-230'** : C in RCH2NRR H on C is type 140
- **'CT-231'** : C in RN(CH3)2 H on C is type 140
- **'C -235'** : C: C=O in amide. Acyl R in amides is neutral - use alkane parameters.
- **'O -236'** : O: C=O in amide. Acyl R in amides is neutral - use alkane parameters.
- **'N -237'** : N: primary amide.
- **'N -238'** : N: secondary amide
- **'N -239'** : N: tertiary amide
- **'H -240'** : H on N: primary amide
- **'H -241'** : H on N: secondary amide
- **'CT-242'** : C on N: secondary N-Me amide
- **'CT-243'** : C on N: tertiary N-Me amide
- **'CT-244'** : C on N: secondary N-CH2R amide
- **'CT-245'** : C on N: tertiary N-CH2R amide (Pro Cdelta)
- **'CT-246'** : C on N: tertiary N-CHR2 amide (Pro Calpha)
- **'C -247'** : C in O=C(NH2)2 Urea
- **'O -248'** : O in O=C(NHw)w Urea Isr. J. Chem. 33, 323 (93)
- **'N -249'** : N in O=C(NH2)2 Urea Isr. J. Chem. 33, 323 (93)
- **'H -250'** : H in O=C(NH2)2 Urea
- **'N -251'** : N in imide
- **'C -252'** : C(=O) in imide
- **'O -253'** : O in imide
- **'H -254'** : H(N) in imide
- **'HC-255'** : H(C) in formimide
- **'CT-256'** : C in CH3 imide
- **'CT-257'** : C in RCH2 imide
- **'CT-258'** : C in R2CH imide
- **'CT-259'** : C in R3C imide
- **'C -267'** : Co in CCOOH carboxylic acid
- **'OH-268'** : Oh in CCOOH: R in RCOOH is neutral; use 135-140
- **'O -269'** : Oc in CCOOH
- **'HO-270'** : H in CCOOH
- **'C -271'** : C in COO- carboxylate

- **'O2-272'** : O: O in COO- carboxylate
- **'CT-273'** : C: CH3, carboxylate ion
- **'CT-274'** : C: CH2, carboxylate ion
- **'CT-275'** : C: CH, carboxylate ion
- **'CT-276'** : C: C, carboxylate ion
- **'C -277'** : AA C: aldehyde - for C-alpha use 135-139
- **'O -278'** : AA O: aldehyde
- **'HC-279'** : AA H-alpha in aldehyde & formamide
- **'C -280'** : AA C: ketone - for C-alpha use 135-139
- **'O -281'** : AA O: ketone
- **'HC-282'** : AA H on C-alpha in ketone & aldehyde
- **'N3-286'** : N (NH4+) JPC,90,2174 (1986)
- **'N3-287'** : N (RNH3+) JPC,90,2174 (1986)
- **'H3-289'** : H (NH4+) JPC,90,2174 (1986)
- **'H3-290'** : H (RNH3+) JPC,90,2174 (1986)
- **'CT-291'** : C in CH3NH3+
- **'CT-292'** : C in RCH2NH3+
- **'CT-293'** : C in R2CHNH3+
- **'CT-294'** : C in R3CNH3+
- **'N2-300'** : N: guanidinium NH2
- **'H3-301'** : H: guanidinium NH2
- **'CA-302'** : C: guanidinum C+
- **'N2-303'** : N: guanidinium NHR
- **'H3-304'** : H: guanidinium NHR
- **'CT-305'** : C: CH3, methylguanidinium
- **'CT-306'** : C: CH3, ethylguanidinium
- **'CT-307'** : C: CH2(D), ARG,ethylguanidinium
- **'CT-308'** : C: CH2(G), ARG
- **'NA-311'** : DAP N1 Diamino-pyridine
- **'CA-312'** : DAP C2 Diamino-pyridine
- **'N2-313'** : DAP N-amine
- **'H2-314'** : DAP H-amine
- **'CA-315'** : DAP C3
- **'HC-316'** : DAP H3
- **'CA-317'** : DAP C4
- **'HC-318'** : DAP H4
- **'NA-319'** : Uracil N1 - use AT=N* for nucleoside
- **'C -320'** : Uracil C2
- **'NA-321'** : Uracil N3
- **'C -322'** : Uracil C4
- **'CM-323'** : Uracil C5
- **'CM-324'** : Uracil C6
- **'H -325'** : Uracil H-N1
- **'O -326'** : Uracil O-C2
- **'H -327'** : Uracil H-N3
- **'O -328'** : Uracil O-C4
- **'HC-329'** : Uracil H-C5
- **'HC-330'** : Uracil H-C6
- **'CT-331'** : Thymine C-C5
- **'HC-332'** : Thymine H-CC5
- **'NA-333'** : Cytosine N1 - use AT=N* for nucleoside
- **'C -334'** : Cytosine C2
- **'NC-335'** : Cytosine N3
- **'CA-336'** : Cytosine C4 Nucleotide base parameters: JACS 113,2810 (1991)

- **'CA-337'** : Cytosine C5
- **'CM-338'** : Cytosine C6
- **'H -339'** : Cytosine H-N1
- **'O -340'** : Cytosine O-C2
- **'N2-341'** : Cytosine N-C4
- **'H2-342'** : Cytosine H-NC4/N3
- **'H2-343'** : Cytosine H-NC4/C5
- **'HC-344'** : Cytosine H-C5
- **'HC-345'** : Cytosine H-C6
- **'NC-346'** : Adenine N1
- **'CQ-347'** : Adenine C2
- **'NC-348'** : Adenine N3
- **'CB-349'** : Adenine C4
- **'CB-350'** : Adenine C5
- **'CA-351'** : Adenine C6
- **'NB-352'** : Adenine N7 Guanine
- **'CK-353'** : Adenine C8 Guanine
- **'NA-354'** : Adenine N9 Guanine - use AT=N* for nucleoside
- **'HC-355'** : Adenine H-C2
- **'N2-356'** : Adenine N-C6
- **'H2-357'** : Adenine H-NC6/N1
- **'H2-358'** : Adenine H-NC6/C5
- **'HC-359'** : Adenine H-C8 Guanine
- **'H -360'** : Adenine H-N9 Guanine
- **'NA-361'** : Guanine N1
- **'CA-362'** : Guanine C2
- **'NC-363'** : Guanine N3
- **'CB-364'** : Guanine C4
- **'CB-365'** : Guanine C5
- **'C -366'** : Guanine C6
- **'H -367'** : Guanine H-N1
- **'N2-368'** : Guanine N-C2
- **'H2-369'** : Guanine H-NC2
- **'O -370'** : Guanine O-C6
- **'CT-371'** : 9-Me A or G C-N9
- **'HC-372'** : 9-Me A or G H-CN9
- **'CT-373'** : 1-Me U or T C-N1
- **'HC-374'** : 1-Me U or T H-CN1
- **'CT-375'** : 1-Me Cytosine C-N1
- **'HC-376'** : 1-Me Cytosine H-CN1
- **'NA-377'** : CytH+ N1 Use AT=N* for nucleoside.
- **'C -378'** : CytH+ C2
- **'NA-379'** : CytH+ N3 Protonated cytosine.
- **'CA-380'** : CytH+ C4
- **'CM-381'** : CytH+ C5
- **'CM-382'** : CytH+ C6
- **'H -383'** : CytH+ H-N1
- **'O -384'** : CytH+ O-C2
- **'H -385'** : CythH+ H-N3
- **'N2-386'** : CytH+ N-C4
- **'H2-387'** : CytH+ H-NC4/N3
- **'H2-388'** : CytH+ H-NC4/C5
- **'HC-389'** : CytH+ H-C5
- **'HC-390'** : CytH+ H-C6

- **'CT-391'** : 1-Me CytH+ C-N1
- **'HC-392'** : 1-Me CytH+ H-CN1
- **'P -393'** : P dimethylphosphate anion
- **'O2-394'** : O(=) dimethylphosphate anion
- **'OS-395'** : O(-) dimethylphosphate anion
- **'CT-396'** : C in CH3 dimethylphosphate anion
- **'F -400'** : F- JACS 106,903 (1984)
- **'Cl-401'** : Cl- JACS 106, 903 (1984)
- **'Br-402'** : Br- JACS 107, 7793 (1985)
- **'Li-404'** : Li+ JACS 106, 903 (1984)
- **'Na-405'** : Na+ JACS 106, 903 (1984)
- **'Li-406'** : Li+ Aqvist's cation parameters: JPC,94,8021 (90)
- **'Na-407'** : Na+ Aqvist's cation parameters: JPC,94,8021 (90)
- **'K -408'** : K+ Aqvist's cation parameters: JPC,94,8021 (90)
- **'Rb-409'** : Rb+ Aqvist's cation parameters: JPC,94,8021 (90)
- **'Cs-410'** : Cs+ Aqvist's cation parameters: JPC,94,8021 (90)
- **'Mg-411'** : Mg++ Aqvist's cation parameters: JPC,94,8021 (90)
- **'Ca-412'** : Ca++ Aqvist's cation parameters: JPC,94,8021 (90)
- **'Sr-413'** : Sr++ Aqvist's cation parameters: JPC,94,8021 (90)
- **'Ba-414'** : Ba++ Aqvist's cation parameters: JPC,94,8021 (90)
- **'O -434'** : O in OH- Hydroxide O-H = 0.953A
- **'HO-435'** : H in OH- JACS 108,2517 (86)
- **'U -436'** : U in UO2+ J. Mol. Struct. 366, 55 (96)
- **'OU-437'** : O in UO2+ r(U-O) = 1.80 A
- **'CA-500'** : CG in Trp
- **'CA-501'** : CD C in Trp
- **'CA-502'** : CE C in Trp
- **'NA-503'** : NE in Trp
- **'H -504'** : H on NE in Trp
- **'CT-505'** : CB in His
- **'CA-506'** : CE1 in HID,HIE
- **'CA-507'** : CD2 in HID, CG in HIE
- **'CA-508'** : CG in HID, CD2 in HIE
- **'CA-509'** : CE1 in HIP
- **'CA-510'** : CG, CD2 in HIP
- **'NB-511'** : NE in HID, ND in HIE
- **'NA-512'** : N in HIP
- **'H -513'** : H on N in HIP

Coulombic interactions

OPLS-1996 uses atom-centered point charges to represent the electrostatic interactions. Each of the atom types listed above has a default charge suggested by the forcefield and you can use these default charges by using the 'bond increment' option.

Improper torsions

OPLS-aa uses stereocenter improper torsions to enforce planarity in aromatic rings, around amide nitrogens, and around other sp2 carbons. Remember that you can set the improper type to 0 to have the code automatically determine the improper type (so long as inpstyle is 2). Improper torsions for proteins are exactly the same as implemented in AMBER96 - even though OPLS puts the stereocenter in the second position of the torsion, you still define them in towhee putting it in the first position (towhee automatically adjusts the parameter order for you).

Proteins

The OPLS-aa protein builder has been implemented with all charges verbatim from WLJ 1996. (Note that the sidechain charges on Met and Cys were later revised in OPLS-2001). Because many of the bonded parameters are unpublished, they have been cobbled together from other OPLS-aa implementations, namely ffoplsaabon.itp in GROMACS 3.2.1 and oplsaa.prm in TINKER 4.0- the origin of added parameters is referenced in the comments of ffoplsaa.F. Most of the missing parameters merely required additional aliases to existing entries because of the above naming convention. Note that GROMACS uses nanometers and KJoules instead of Angstroms and Kcals so unit conversions are necessary when comparing Towhee and GROMACS results. Here is a complete list of the codes for the 20 amino acids, plus some other functional groups that work with the protein builder.

- 'a0' alanine
- 'c0' cysteine with hydrogen on the sulfur
- 'cs' cysteine in a disulfide bond
- 'd-' aspartic acid deprotonated
- 'e-' glutamic acid deprotonated
- 'f0' phenylalanine
- 'g0' glycine
- 'h+' histidine both N protonated
- 'hd' histidine neutral with only N_d protonated
- 'he' histidine neutral with only N_e protonated
- 'i0' isoleucine
- 'k+' lysine protonated
- 'l0' leucine
- 'm0' methionine
- 'n0' asparagine
- 'p0' proline
- 'q0' glutamine
- 'r+' arginine protonated
- 's0' serine
- 't0' threonine
- 'v0' valine
- 'w0' tryptophan. Despite a great deal of effort this amino acid is still not working properly with this forcefield as there are many parameters that are missing. You will get an error if you include this amino acid for this forcefield.
- 'y0' tyrosine
- 'za' acetyl cap on the N-terminus
- 'zm' amide cap (NH_2) on the C-terminus.
- 'zn' N-methylamide cap on the C-terminus
- 'zf' formaldahyde cap on the N-terminus

Return to the Towhee Capabilities web page

Send comments to: Marcus G. Martin
Last updated: July 18, 2006

MCCCS Towhee (OPLS-2001)

Overview

This section covers the OPLS-2001 force field as it is implemented into the towhee_ff_OPLS-2001 file in the ForceFields directory. All of the Towhee atom types for the OPLS-2001 force field are listed, along with a short description of their meanings. For more information about the OPLS force fields see the Jorgensen group home page. Note that OPLS-2001 is a Lennard-Jones (12-6) force field and can only be combined with other Lennard-Jones (12-6) force fields. The rather famous OPLS water models are not included here as they are provided in their own force field files. Please see the TIP3P, TIP4P, and TIP5P web pages for more information about those water models. Note that OPLS-2001 was preceeded by a united-atom version of the force field called OPLS-ua and an earlier version of the all-atom force field called OPLS-aa. I would like to acknowledge W.L. Jorgensen for kindly providing me with an electronic copy of the OPLS-2001 parameters. Any discrepencies (especially typos) from the published OPLS-2001 force field values are the sole responsibility of Marcus G. Martin, and I welcome feedback on how this implementation compares with other programs.

References for OPLS-2001

OPLS-2001 was implemented acording to an electronic file available from the Jorgensen group, but the official reference is also listed below.

- Jorgensen *personal communication* 2001
- Kaminski *et al.* 2001

OPLS-2001 in Towhee

The official force field name for OPLS-2001 in Towhee is 'OPLS-2001'. Here I list all of the OPLS-2001 atom names for use in the towhee_input file, along with a brief description taken from the OPLS-2001 literature. Notice that OPLS-2001 uses almost the same naming conventions as the Amber param96 force field. This is not a coincedence as these force fields utilize many of the same bonded interaction terms. I have modified some of the atom names from the standard OPLS-2001 in order to create unique type names. This was needed because OPLS-2001 occasionally uses different bonded interactions for atoms which have the same atom name for nonbonded interactions. I do not always distinguish between the original OPLS-2001 comments, and my modifications as I have created several new atom names. Some of my comments are placed in [square brackets]. Please note that the capitalization and spacing pattern is important and must be followed exactly as listed here.

Bromine

- **'Br-'** : bromine ion (charge -1)[uses the parameters from the LGM force field]

Carbon

- **'C a'** : carbon in O-C=O acid
- **'C k'** : carbon in C=O not bonded to N, and not an acid group
- **'C n'** : carbon in N-C=O
- **'C*'** : aromatic C in 5-membered ring next to two carbons
- **'CA~'** : neutral aromatic carbon not at the junction of 5-membered and 6-membered rings
- **'CA+'** : aromatic carbon in guanidinium C+
- **'CAj'** : neutral aromatic carbon at the juction of 5-membered and 6-membered rings
- **'CAxna'** : special parameters for nucleotide bases. aromatic C in DAP C2,C3,C4 pyridine; Cytosine C4,C5,C6
- **'CBxna'** : special parameters for nucleotide bases. Adenine C4,C5,C6; Guanine

C2,C4,C5

- **'CB'** : aromatic C at juntion of 5-membered and 6-membered rings
- **'CKxna'** : special parameters for nucleotide bases. Adenine C8 Guanine
- **'C='** : alkene sp^2 carbon (non-aromatic) interior carbons in a diene.
- **'CM'** : alkene sp^2 carbon (non-aromatic)
- **'CMxna'** : special parameters for nucleotide bases. Carbon in Uracil C5,C6
- **'CO'** : sp^3 anomeric carbon (bonded to ether O and alcohol O). this type shows up in carbohydrates
- **'CQ'** : sp^{2}C in 6-membered ring between deprotonated N's
- **'CR'** : aromatic C in 5-membered ring next to two nitrogens
- **'CS'** : aromatic C which does not have 120 degree bond angles with all of its neighbors. Examples are C3 in pyrrole and C3 in furan
- **'CT'** : aliphatic sp^3 hybrid carbon
- **'CTf'** : aliphatic sp^3 hybrid carbon bonded to F (monoalkyl fluorides and perfluoroalkanes)
- **'CTcf4'** : aliphatic sp^3 hybrid carbon bonded to F in CF4
- **'CTxna'** : special parameters for nucleotide bases. Thymine C-C5, 9-Me A or G C-N9
- **'CU'** : aromatic C which abuts NB in a heteroatom-heteroatom bond. Examples are C3 in pyrazole and C3 in isoxazole
- **'CV'** : aromatic C in 5-membered ring next to C and deprotonated N
- **'CW'** : sp^2 aromatic C in 5-membered ring next to C and NH

Chlorine

- **'Cl'** : chlorine bonded to a carbon
- **'Cl-'** : chlorine ion (charge -1)

Fluorine

- **'F'** : fluorine nonionic (monoalkyl fluorides)
- **'Fpf'** : fluorine nonionic (perfluoroalkanes)
- **'F-'** : fluorine ion (charge -1)

Hydrogen

- **'H'** : H attached to N
- **'H2xna'** : special parameters for nucleotide bases. DAP H-amine, Cytosine, Adenine
- **'HA'** : H bonded to an aromatic ring
- **'HC~1'** : hydrogen attached to an sp^3 carbon in most cases, see special cases of hydrogen below
- **'HC~2'** : H bonded to a non-aromatic sp^2 carbon which is not bonded to a N, not double bonded to O, and is not bonded to C=O
- **'HC~3'** : H bonded to a C=O or bonded to a C which is bonded to a C=O
- **'HC~4'** : H bonded to a CT that is bonded to NT
- **'HCxna'** : special parameters for nucleotide bases. DAP H3,H4; Uracil H-C5,H-C6; Thymine H-CC5; Cytosine H-C5,H-C6; Adenine H-C2,H-C8; Guanine
- **'HO'** : H attached to O in alcohol
- **'HS'** : H attached to S
- **'HW'** : H on TIP3P, TIP4P or TIP5P water

Iodine

- **'I-'** : iodine ion (charge -1)

Lithium

- **'Li+'** : lithium ion (charge +1)

Nitrogen

- **'N'** : N in an amide
- **'N2'** : sp^2 N of aromatic amines and guanidinium ions
- **'NA'** : sp^2 aromatic N with H attached

- **'NB'** : sp^2 N in 5-membered ring, deprotonated
- **'NC'** : sp^2 N in 6-membered ring, deprotonated
- **'NO'** : sp^2 N in a nitro group
- **'NT~1'** : sp^3 N in ammonia
- **'NT+1'** : sp^3 N in ammonium ion (charge +1)
- **'NT~2'** : sp^3 N in primary, secondary, tertiary amine
- **'NT+2'** : sp^3 N in primary, secondary, tertiary ammonium ion (charge +1)

Oxygen

- **'O'** : oxygen in C=O, not an acidic site
- **'O2'** : oxygen double bonded to carbon in COO- or COOH
- **'OHa'** : oxygen bonded to H in RCOOH
- **'OHm'** : oxygen bonded to H in a mono-alcohol
- **'OHp'** : oxygen bonded to H in polyols or phenol
- **'OHd'** : oxygen bonded to H in diols
- **'OHt'** : oxygen bonded to H in triols
- **'OS'** : sp^3 O in an ether or acetal
- **'OWt3p'** : water oxygen in the TIP3P model
- **'OWt4p'** : water oxygen in the TIP4P model
- **'OWt5p'** : water oxygen in the TIP5P model
- **'L4p'** : negative charge site on the TIP4P water
- **'L5p'** : negative charge sites on the TIP5P water

Sodium

- **'Na+'** : sodium ion (charge +1)

Sulfur

- **'S'** : S in sulfide, disulfide
- **'SH'** : S in thiols

Coulombic interactions

OPLS-2001 uses atom-centered point charges to represent the electrostatic interactions. I do not know of an automated way to assign these point charges. Instead, you need to consult the OPLS-2001 literature, or (even better) ask the Jorgensen group for a copy of the force field file, and then find molecules with similar moieties to the ones on the molecule you wish to simulate.

Improper torsions

OPLS-2001 uses stereocenter improper torsions to enforce planarity in aromatic rings, around amide nitrogens, and around other sp2 carbons. Remember that you can set the improper type to 0 to have the code automatically determine the improper type (so long as inpstyle is 2).

Proteins

OPLS-2001 has a complete parameter set for proteins, but I have not yet implemented this into the protein builder. It is on my list of things to do, but is not a high priority at the moment. If someone else would like to add it into the builder I would be happy to provide advice about how to perform that task. Note that since the OPLS-aa protein builder is implemented, only minor changes would be needed to implement OPLS 2001 (the primary difference is the amino-acid-specific torsions).

Return to the Towhee Capabilities web page

Send comments to: Marcus G. Martin
Last updated: September 07, 2005

MCCCS Towhee (Panagiotopoulos 1989)

Overview

This section covers the Panagiotopoulos 1989 (Pana1989) force field as it is implemented into the towhee_ff_Pana1989 file in the ForceFields directory. This force field was designed for vapor-liquid coexistence of Noble gasses. Note that this is a Lennard-Jones force field that ended up using an explicit mixing rule between certain (but not all) pairs of atoms. The 'Shukla' mixing rule will reproduce the cross terms they used for Argon-Krypton. However, it can also be used with any of the normal Lennard-Jones mixing rules. Any discrepencies (especially typos) from the published values are the sole responsibility of Marcus G. Martin, and I welcome feedback on how this implementation compares with other programs.

References for Pana1989

This forcefield is described in the following paper.

- Panagiotopoulos 1989
  And the atomic polarizabilites are taken from Table D.3 of
- Gray and Gubbins 1984

Pana1989 in Towhee

The official force field name for Panagiotopoulos 1989 in Towhee is 'Pana1989'. Here is the list of atom names for use in the towhee_input file, along with a brief description. Please note that the capitalization and spacing pattern is important and must be followed exactly as listed here.

- **'Ne'** : neon
- **'Ar'** : argon
- **'Kr'** : krypton
- **'Xe'** : xenon

Coulombic interactions

This force field does not include Coulombic interactions.

Return to the Towhee Capabilities web page

Send comments to: Marcus G. Martin
Last updated: April 18, 2006

MCCCS Towhee (Potter *et al.* 1997)

Overview

This section covers the Potter *et al.* 1997 (Potter1997) force field as it is implemented into the towhee_ff_Potter1997 file in the ForceFields directory. This force field was designed to reproduce the vapour-liquid equilibria for three different fluoromethane molecules. Note that this is a Lennard-Jones force field that uses the 'LB plus manual' mixing rules, and therefore you will need to explicitly set some cross terms in towhee_input in order to use this force field as the authors intended. Any discrepencies (especially typos) from the published values are the sole responsibility of Marcus G. Martin, and I welcome feedback on how this implementation compares with other programs.

References for Potter *et al.* 1997

This forcefield is described in a single paper.

- Potter *et al.* 1997

Typos and comments for Potter *et al.* 1997

- Please note that you need to explicitly set the cross terms for H and F in towhee_input in order to properly reproduce their work. The cross terms for the H-F interactions should have a sigma of 2.3 Angstrom and an epsilon of 22.36 Kelvin. See the Potter_CF2H2 example for more information about how to set these cross terms.

Potter *et al.* 1997 in Towhee

The official force field name for Potter *et al.* 1997 in Towhee is 'Potter1997'. Here is the list of atom names for use in the towhee_input file, along with a brief description. I created these names in order to distinguish between the different hydrogen and fluorine parmeters that are used depending upon the extent of methane fluorination. Please note that the capitalization and spacing pattern is important and must be followed exactly as listed here.

- **'C'** : Carbon in any of the fluoromethanes
- **'F(2)'** : Fluorine in CH_2F_2
- **'F(3)'** : Fluorine in CHF_3
- **'F(4)'** : Fluorine in CF_4
- **'H(2)'** : Hydrogen in CH_2F_2
- **'H(1)'** : Hydrogen in CHF_3

Coulombic interactions

This force field uses point charges and has been set up to assign the point charges using the bond increment method. The charges for bmim are assigned according to Table 1 of Potter *et al.* 1997.

Return to the Towhee Capabilities web page

Send comments to: Marcus G. Martin
Last updated: April 27, 2006

MCCCS Towhee (Shah and Maginn 2004)

Overview

This section covers the Shah and Maginn 2004 (Shah2004) force field as it is implemented into the towhee_ff_Shah2004 file in the ForceFields directory. This force field was designed for an ionic liquid of 1-*n*-butyl-3-methylimidazolium [bmim] hexafluorophosphate [PF_6]. Note that this is a Lennard-Jones force field that uses the 'Lorentz-Berthelot' mixing rules. Any discrepencies (especially typos) from the published values are the sole responsibility of Marcus G. Martin, and I welcome feedback on how this implementation compares with other programs.

References for Shah and Maginn 2004

This forcefield is described in a single paper.

- Shah and Maginn 2004

Typos and comments for Shah and Maginn 2004

- Full Lennard-Jones interactions are used for 1-4 interactions, while the Coulombic 1-4 terms are scaled by 1/2 [personal communication between E.J. Maginn and M.G. Martin 04-21-2006]. This is with the execption of the ring dihedrals that are held rigid and so there is no reason to compute the 1-4 terms.
- Note that the coordinates presented for the PF_6 structure result in asymmetric bond lengths and bending angles. I have inlcluded that as an atom-specific implementation of PF_6, and I have also included an option for symmetric PF_6 using the bond length from Morrow and Maginn 2002.
- Note that all of the rigid interactions in this molecule means you need to start the simulation using a template for both molecules. Coordinates suitable for a template are found in Table 1 of Shah and Maginn 2004.

Shah and Maginn 2004 in Towhee

The official force field name for Shah and Maginn 2004 in Towhee is 'Shah2004'. Here is the list of atom names for use in the towhee_input file, along with a brief description. These names correspond to those used in Shah and Maginn 2004. I have also included an option for a symmetric PF_6 using a perfect octahedral geometry and a P-F bond length taken from Morrow and Maginn 2002. Please note that the capitalization and spacing pattern is important and must be followed exactly as listed here.

- **'N1'** : Ring Nitrogen bonded to C2, C5, and C7 in bmim
- **'C2'** : Ring Carbon (and an implicit bonded hydrogen) bonded to N1 and N3 in bmim
- **'N3'** : Ring Nitrogen bonded to C2, C4, and C6 in bmim
- **'C4'** : Ring Carbon (and an implicit bonded hydrogen) bonded to N3 and C5 in bmim
- **'C5'** : Ring Carbon (and an implicit bonded hydrogen) bonded to N1 and C4 in bmim
- **'C6'** : methyl Carbon (and three implicit bonded hydrogens) bonded to N3 in bmim
- **'C7'** : methylene Carbon (and two implicit bonded hydrogens) bonded to N1 and C8 in bmim
- **'C8'** : methylene Carbon (and two implicit bonded hydrogens) bonded to C7 and C9 in bmim
- **'C9'** : methylene Carbon (and two implicit bonded hydrogens) bonded to C8 and C10 in bmim
- **'C10'** : methyl Carbon (and three implicit bonded hydrogens) bonded to C9 in bmim
- **'P'** : P in hexafluorophosphate

'F' : generic F in hexafluorophosphate if you want to use the symmetric bond length and bending angle version of that molecule.

- **'F1'** : specific F in hexafluorophosphate if you want to use the assymmetric bond length and bending angle version of that molecule.
- **'F2'** : specific F in hexafluorophosphate if you want to use the assymmetric bond length and bending angle version of that molecule.
- **'F3'** : specific F in hexafluorophosphate if you want to use the assymmetric bond length and bending angle version of that molecule.
- **'F4'** : specific F in hexafluorophosphate if you want to use the assymmetric bond length and bending angle version of that molecule.
- **'F5'** : specific F in hexafluorophosphate if you want to use the assymmetric bond length and bending angle version of that molecule.
- **'F6'** : specific F in hexafluorophosphate if you want to use the assymmetric bond length and bending angle version of that molecule.

Coulombic interactions

This force field uses point charges and has been set up to assign the point charges using the bond increment method. The charges for bmim are assigned according to Table 2 of Shah and Maginn 2004.

Return to the Towhee Capabilities web page

Send comments to: Marcus G. Martin
Last updated: April 27, 2006

MCCCS Towhee (Shukla 1987)

Overview

This section covers the Shukla 1987 (Shukla1987) force field as it is implemented into the towhee_ff_Shukla1987 file in the ForceFields directory. This force field was designed for multicomponent fluid mixtures of light gasses. Note that this is a Lennard-Jones force field that uses the 'Shukla' mixing rules. However, it can also be used with any of the normal Lennard-Jones mixing rules. Any discrepencies (especially typos) from the published values are the sole responsibility of Marcus G. Martin, and I welcome feedback on how this implementation compares with other programs.

References for Shukla1987

This forcefield is described in the following paper.

- Shukla 1987

Shukla 1987 in Towhee

The official force field name for Shukla 1987 in Towhee is 'Shukla1987'. Here is the list of atom names for use in the towhee_input file, along with a brief description. Please note that the capitalization and spacing pattern is important and must be followed exactly as listed here.

- **'Ar'** : argon
- **'Kr'** : krypton
- **'Xe'** : xenon
- **'CH4'** : methane
- **'N2'** : nitrogen dimer

Coulombic interactions

This force field does not include Coulombic interactions.

Return to the Towhee Capabilities web page

Send comments to: Marcus G. Martin
Last updated: April 18, 2006

MCCCS Towhee (SKS)

Overview

This section covers the Smit-Karaborni-Siepmann Force Field (SKS) for n-alkanes as it is implemented into the towhee_ff_SKS file in the ForceFields directory. All of the Towhee atom types for this force field are listed, along with a short description of their meanings. SKS uses a 'Lennard-Jones' classical_potential and the 'Lorentz-Berthelot' classical_mixrule. This forcefield is best used with a 13.8 Angstrom cutoff with no tail corrections and without shifting at the cutoff. This strange set of parameters is due to a bug in the code originally used to produce this forcefield. Please see the Erratum in the references section for more information. The SKS forcefield was later extended to branched alkanes (SMMK Main) and eventually replaced by the TraPPE-UA force field. Any discrepencies (especially typos) from the published force field values are the sole responsibility of Marcus G. Martin, and I welcome feedback on how this implementation compares with other programs.

References for SKS

The parameters and detailed results for the SKS force field are found here.

- Smit *et al.* 1995

Please note that an error was made in the code that produced the results in that paper that essentially meant that tail corrections were not employed. Please see the following erratum for more information.

- Smit *et al.* Erratum 1998

SKS in Towhee

The official force field name for SKS in Towhee is 'SKS'. Here I list all of the atom names for use in the towhee_input file, along with a brief description. Please note that the capitalization and spacing pattern is important and must be followed exactly as listed here.

- **'CH3'** methyl group (C plus 3 bonded H as a united-atom) on an *n*-alkane of C_4 or longer.
- **'CH2'** methylene group (C plus 2 bonded H as a united-atom) on an *n*-alkane of C_4 or longer.

Coulombic interactions

There are no coulombic interactions for this force field.

Return to the Towhee Capabilities web page

Send comments to: Marcus G. Martin
Last updated: September 07, 2005

MCCCS Towhee (Smith and Dang 1994)

Overview

This section covers the Smith and Dang 1994 non-polarizable parameters for NaCl in SPC/E water as they are implemented into the towhee_ff_Smith1994 file in the ForceFields directory. All of the Towhee atom types for this force field are listed, along with a short description of their meanings. Note that this is a Lennard-Jones (12-6) force fields that use Geometric mixing rules and can only be combined with other Lennard-Jones (12-6) force fields. It was designed to be used in combination with the SPC/E water model. Any discrepencies (especially typos) from the published force field values are the sole responsibility of Marcus G. Martin, and I welcome feedback on how this implementation compares with other programs.

References for Smith and Dang 1994

- Smith and Dang 1994

Smith and Dang 1994 in Towhee

The official force field name for Smith and Dang 1994 in Towhee is 'Smith1994'. Here I list all of the atom names for use in the towhee_input file, along with a brief description. The meanings should be quite obvious for this force field. Please note that the capitalization and spacing pattern is important and must be followed exactly as listed here.

- **'Na'** : sodium ion with a +1 charge.
- **'Cl'** : chloride ion with a -1 charge.

Coulombic interactions

This force field uses atom-centered point charges to represent the electrostatic interactions. The 'bond increment' method for assigning charges is implemented for these molecules. Please see the inpstyle 2 documentation for further information on enabling this option. As there are no bonds this option will simply assign the default ionic charges of +1 to sodium and -1 to chloride.

Return to the Towhee Capabilities web page

Send comments to: Marcus G. Martin
Last updated: September 07, 2005

MCCCS Towhee (SMMK Main)

Overview

This section covers the Siepmann-Martin-Mundy-Klein (SMMK) force field for branched alkanes as it is implemented into the towhee_ff_SMMKmain file in the ForceFields directory. All of the Towhee atom types for this force field are listed, along with a short description of their meanings. SMMKmain uses a 'Lennard-Jones' classical_potential and the 'Geometric' classical_mixrule. Unlike its predecessor (SKS) the SMMKmain force field uses tail corrections, and was originally used with a 13.8 Angstrom cutoff. The SMMKmain forcefield is an extension of the SKS force field for linear alkanes and was immediately supplanted by similar parameters included in the same paper (see SMMK Note Added in Proof and eventually replaced by the TraPPE-UA force field. Any discrepencies (especially typos) from the published force field values are the sole responsibility of Marcus G. Martin, and I welcome feedback on how this implementation compares with other programs.

References for SMMK Main

Most of the nonbonded parameters and detailed results for the SMMK Main force field are found in the main text of the following paper.

- Siepmann *et al.* 1997

This is a fully flexible model where the intramolecular terms for all of the atoms, and some special One-Five nonbonded terms for **CH3methyl** groups are taken from.

- Mundy *et al.* 1996

SMMK Main in Towhee

The official force field name for SMMK Main in Towhee is 'SMMKmain'. Here I list all of the atom names for use in the towhee_input file, along with a brief description. Please note that the capitalization and spacing pattern is important and must be followed exactly as listed here.

- **'CH3long'** methyl group (C plus 3 bonded H as a united-atom) on an *n*-alkane of C_4 or longer, or on an alkyl side-chain of length propyl or longer.
- **'CH3ethyl'** methyl group (C plus 3 bonded H as a united-atom) on an ethyl side-chain.
- **'CH3methyl'** methyl group (C plus 3 bonded H as a united-atom) on an methyl side-chain.
- **'CH2'** methylene group (C plus 2 bonded H as a united-atom).
- **'CH'** methine group (C plus 1 bonded H as a united-atom).
- **'C'** quaternary carbon.

Coulombic interactions

There are no coulombic interactions for this force field.

Return to the Towhee Capabilities web page

Send comments to: Marcus G. Martin
Last updated: November 10, 2005

MCCCS Towhee (SMMK Note added in proof)

Overview

This section covers the Siepmann-Martin-Mundy-Klein (SMMK) force field for branched alkanes as it is implemented into the towhee_ff_SMMKnaip file in the ForceFields directory. All of the Towhee atom types for this force field are listed, along with a short description of their meanings. SMMKnaip uses a 'Lennard-Jones' classical_potential and the 'Lorentz-Berthelot' classical_mixrule. The SMMKnaip force field was originally used with a 14.0 Angstrom cutoff and tail corrections. The SMMKnaip forcefield is an improvement upon the SMMK Main force field whose parameters were listed in the main text of the same paper. SMMKnaip was a predecessor of the TraPPE-UA force field. Any discrepencies (especially typos) from the published force field values are the sole responsibility of Marcus G. Martin, and I welcome feedback on how this implementation compares with other programs.

References for SMMK Note added in proof

The parameters and detailed results for the SMMK Note added in proof force field are found in the note added in proof of the following paper.

- Siepmann *et al.* 1997

The angle and torsion terms for this force field are taken from the following paper.

- Mundy *et al.* 1996

Note that unlike the SMMK Main implementation, this forcefield uses rigid bond lengths.

SMMK Note added in proof in Towhee

The official force field name for SMMK Note added in proof in Towhee is 'SMMKnaip'. Here I list all of the atom names for use in the towhee_input file, along with a brief description. Please note that the capitalization and spacing pattern is important and must be followed exactly as listed here.

- **'CH3'** methyl group (C plus 3 bonded H as a united-atom).
- **'CH2'** methylene group (C plus 2 bonded H as a united-atom).
- **'CH'** methine group (C plus 1 bonded H as a united-atom).

Coulombic interactions

There are no coulombic interactions for this force field.

Return to the Towhee Capabilities web page

Send comments to: Marcus G. Martin
Last updated: September 07, 2005

MCCCS Towhee (SPC-E water)

Overview

This section covers the Simple Point Charge - Extended (SPC-E) water model parameters as they are implemented into the towhee_ff_SPC-E file in the ForceFields directory. All of the Towhee atom types for the SPC-E water force field are listed, along with a short description of their meanings. This is not really a comphrehensive stand-alone force field, but instead a popular model for water. Note that these are Lennard-Jones (12-6) parameters and can only be combined with other Lennard-Jones (12-6) force fields. Any discrepencies (especially typos) from the published SPC-E force field values are the sole responsibility of Marcus G. Martin, and I welcome feedback on how this implementation compares with other programs.

References for SPC-E water

- Berendsen *et al.* 1987

SPC-E in Towhee

The official force field name for Simple Point Charge - Extended water in Towhee is 'SPC-E'. Here I list all of the SPC-E atom names for use in the towhee_input file, along with a brief description (although the meaning of the atom names should be obvious). Please note that the capitalization and spacing pattern is important and must be followed exactly as listed here.

- **'HW'** : hydrogen in water
- **'OW'** : oxygen in water

Coulombic interactions

Atom centered point charges are used to represent the electrostatic interactions.

- **'HW'** : 0.4238
- **'OW'** : -0.8476

Improper torsions

This force field only has parameters for water so there are no improper torsions.

Proteins

This force field only has parameters for water, and while you are welcome to use them with another protein force field, there are no amino acids in the SPC-E force field.

Return to the Towhee Capabilities web page

Send comments to: Marcus G. Martin
Last updated: September 07, 2005

MCCCS Towhee (Sum *et al.* 2003)

Overview

This section covers the Sum *et al.* 2003 United Atom (Sum2003) force field as it is implemented into the towhee_ff_Sum2003 file in the ForceFields directory. This forcefield is an extension to the NERDv2 force field that allows the simulation of acetates and glycerols. I would like to acknowledge Mary Biddy for providing useful guidence about implementing this force field. Any discrepencies (especially typos) from the published values are the sole responsibility of Marcus G. Martin, and I welcome feedback on how this implementation compares with other programs.

References for Sum *et al.* 2003

See the NERD Version 2 force field documentation for the references for the alkane chains. The extensions to that force field are published in a single paper.

- Sum *et al.* 2003

Sum2003 in Towhee

The official force field name for Sum *et al.* United Atom in Towhee is 'Sum2003'. Here I list all of the atom names for use in the towhee_input file, along with a brief description. I created these atom names in order to work with the molecule assembler. Sum2003 is a united-atom force field which lumps hydrogens onto their neighboring carbons. Note that hydrogens that are bonded to atoms other than carbon are not lumped onto their neighboring atoms. The general pattern of the names is a list of the atoms that make up the group, followed by the hybridization of the central atom, and ending in a character string that distinguishes similar atoms. Please note that the capitalization and spacing pattern is important and must be followed exactly as listed here.

sp^3 hybridized Carbons

- **'CH3sp3oac'** : sp^3 carbon plus the 3 hydrogens bonded to it (united-atom) bonded to an sp^3 oxygen in an acetate group.
- **'CH3sp3cac'** : sp^3 carbon plus the 3 hydrogens bonded to it (united-atom) bonded to an sp^2 carbon in an acetate group.
- **'CH3sp3mesc'** : sp^3 carbon plus the 3 hydrogens bonded to it (united-atom) bonded to a CH group (methyl side-chain) and not a special case listed above.
- **'CH3sp3etsc'** : sp^3 carbon plus the 3 hydrogens bonded to it (united-atom) bonded to a CH_2 group that is itself bonded to a CH group (ethyl side-chain).
- **'CH3sp3gen'** : sp^3 carbon plus the 3 hydrogens bonded to it (united-atom) and not one of the cases listed above.
- **'CH2sp3oac'** : sp^3 carbon plus the 2 hydrogens bonded to it (united-atom) that is bonded to an sp^3 oxygen in a glycerol.
- **'CH2sp3cac'** : sp^3 carbon plus the 2 hydrogens bonded to it (united-atom) that is bonded to an sp^2 carbon in an acetate group.
- **'CH2sp3'** : sp^3 carbon plus the 2 hydrogens bonded to it (united-atom) and not a special case.
- **'CHsp3oac'** : sp^3 carbon plus the 1 hydrogen bonded to it (united-atom) that is bonded to an sp^3 oxygen in a glycerol.

3

'CHsp3' : sp carbon plus the 1 hydrogen bonded to it (united-atom) and not a special case.

- **'Csp3'** : sp^3 carbon that is not bonded to any hydrogens.

sp^2 hybridized Carbons

- **'Csp2ac'** : sp^2 carbon double bonded to an sp^2 oxygen, single bonded to an sp^3 oxygen, and single bonded to another carbon in an acetate group.

Oxygen

- **'Osp3'** : sp^3 oxygen bonded to two atoms.
- **'Osp2'** : sp^2 oxygen double bonded to a single atom.

Coulombic interactions

Here is the list of charge assignments for methylacetate. It appears that these are also the charges used for an acetate headgroup on other molecules (e.g. glycerols).

- **'CH3sp3oac'** : 0.200
- **'Osp3'** : -0.400
- **'Csp2ac'** : 0.650
- **'Osp2'** : -0.500
- **'CH3sp3cac'** : 0.050

Improper torsions

This force field does not utilize improper torsions.

Proteins

This force field does not have all of the groups needed for protein simulations.

Return to the Towhee Capabilities web page

Send comments to: Marcus G. Martin
Last updated: September 07, 2005

MCCCS Towhee (Teleman *et al.* 1987)

Overview

This section covers the Teleman *et al.* (Tele1987) force field as it is implemented into the towhee_ff_Tele1987 file in the ForceFields directory. This force field has a flexible implementation of the SPC water model. Note that this is a Lennard-Jones force field that uses either Lorentz-Berthelot or Geometric mixing rules so it is easily combined with other force fields. Any discrepencies (especially typos) from the published values are the sole responsibility of Marcus G. Martin, and I welcome feedback on how this implementation compares with other programs.

References for Tele1987

What I term Tele1987 is described in the following paper.

- Teleman *et al.* 1987

Tele1987 in Towhee

The official force field name for Teleman *et al.* 1987 in Towhee is 'Tele1987'. Here I list all of the atom names for use in the towhee_input file, along with a brief description. I created these atom names in order to work with the molecule assembler. Please note that the capitalization and spacing pattern is important and must be followed exactly as listed here.

- **'H'** : hydrogen in water
- **'O'** : oxygen in water

Coulombic interactions

This force field assigns coulombic interactions to the atoms. The 'bond increment' method of **charge_assignment** is implemented for this forcefield. Please see the inpstyle 2 documentation for further information on enabling this option. Otherwise, you are welcome to manually set the charges. Here I list the bond increments using the atom names listed above. The value of the bond increment is added to the first atom listed and subtracted from the second atom listed.

- H - O: 0.41

Return to the Towhee Capabilities web page

Send comments to: Marcus G. Martin
Last updated: September 07, 2005

MCCCS Towhee (TIP3P)

Overview

This section covers the TIP3P force field for water as it is implemented into the towhee_ff_TIP3P file in the ForceFields directory. All of the Towhee atom types for this force field are listed, along with a short description of their meanings. This is perhaps the most popular water potential of all time and several other force fields use similar parameterizations. TIP3P is a rigid water model that has a Lennard-Jones site on the Oxygen and bare charge sites on the Hydrogens. The Jorgensen group has also parameterized two other water models that utilize additional interaction sites. Please see the TIP4Pand TIP5P web pages for more information about those water models. This water model is suitable for use with any Lennard-Jones forcefield, but is especially well suited for the OPLS family of potentials. Please see the OPLS-ua, OPLS-aa, and OPLS-2001 web pages for more information. Any discrepencies (especially typos) from the published force field values are the sole responsibility of Marcus Martin, and I welcome feedback on how this implementation compares with other programs.

References for TIP3P

TIP3P was originally published in the following paper

- Jorgensen *et al.* 1983

TIP3P in Towhee

The official force field name for TIP3P in Towhee is 'TIP3P'. Here I list all of the atom names for use in the towhee_input file, along with a brief description taken from the literature. Please note that the capitalization and spacing pattern is important and must be followed exactly as listed here.

- **'H'** : hydrogen in water
- **'O'** : oxygen in water

Coulombic interactions

TIP3P uses atom-centered point charges to represent the electrostatic interactions. The 'bond increment' method for assigning charges is implemented for this force field. Please see the inpstyle 2 documentation for further information on enabling this option. Here I list the bond increments for the atom names listed above. The value of the bond increment is added to the first atom listed and subtracted from the second atom listed.

- H - O: 0.417

Return to the Towhee Capabilities web page

Send comments to: Marcus G. Martin
Last updated: September 07, 2005

MCCCS Towhee (TIP4P)

Overview

This section covers the TIP4P force field for water as it is implemented into the towhee_ff_TIP4P file in the ForceFields directory. All of the Towhee atom types for this force field are listed, along with a short description of their meanings. TIP4P is a rigid water model that has a Lennard-Jones site on the Oxygen and bare charge sites on the Hydrogens and along the bisector between the hydrogens. TIP4P is a branched, rigid molecule (when you consider the charged interaction site) and therefore configurational-bias is currently unable to generate structures so you will need to provide an initial structure and also use the rotational-bias moves instead of the configurational-bias moves for molecule transfers between simulation boxes. The Jorgensen group has also parameterized two other water models that utilize different numbers of interaction sites. Please see the TIP3Pand TIP5P web pages for more information about those water models. This water model is suitable for use with any Lennard-Jones forcefield, but is especially well suited for the OPLS family of potentials. Please see the OPLS-ua, OPLS-aa, and OPLS-2001 web pages for more information. Any discrepencies (especially typos) from the published force field values are the sole responsibility of Marcus Martin, and I welcome feedback on how this implementation compares with other programs.

References for TIP4P

TIP4P was originally published in the following paper

- Jorgensen *et al.* 1983

TIP4P in Towhee

The official force field name for TIP4P in Towhee is 'TIP4P'. Here I list all of the atom names for use in the towhee_input file, along with a brief description taken from the literature. Please note that the capitalization and spacing pattern is important and must be followed exactly as listed here.

- **'H'** : hydrogen in water
- **'O'** : oxygen in water
- **'M'** : charge site 0.15 Angstroms along the bisector between the hydrogens

Coulombic interactions

TIP4P uses atom-centered point charges to represent the electrostatic interactions. Here are the suggested charge assignments for water.

- **'H'**: 0.52
- **'O'**: 0.0
- **'M'**: -1.04

Improper torsions

There are no improper torsions for this force field.

Return to the Towhee Capabilities web page

Send comments to: Marcus G. Martin
Last updated: September 07, 2005

MCCCS Towhee (TIP5P)

Overview

This section covers the TIP5P force field for water as it is implemented into the towhee_ff_TIP5P file in the ForceFields directory. All of the Towhee atom types for this force field are listed, along with a short description of their meanings. TIP5P is a rigid water model that has a Lennard-Jones site on the Oxygen and bare charge sites on the Hydrogens and on the lone electron pairs. TIP5P is a branched, rigid molecule (when you consider the lone pairs) and therefore configurational-bias is currently unable to generate structures so you will need to provide an initial structure and also use the rotational-bias moves instead of the configurational-bias moves for molecule transfers between simulation boxes. The Jorgensen group has also parameterized two other water models that utilize different numbers of interaction sites. Please see the TIP3Pand TIP4P web pages for more information about those water models. This water model is suitable for use with any Lennard-Jones forcefield, but is especially well suited for the OPLS family of potentials. Please see the OPLS-ua, OPLS-aa, and OPLS-2001 web pages for more information. Any discrepencies (especially typos) from the published force field values are the sole responsibility of Marcus Martin, and I welcome feedback on how this implementation compares with other programs.

References for TIP5P

TIP5P was originally published in the following paper

- Mahoney and Jorgensen 2000

TIP5P in Towhee

The official force field name for TIP5P in Towhee is 'TIP5P'. Here I list all of the atom names for use in the towhee_input file, along with a brief description taken from the literature. Please note that the capitalization and spacing pattern is important and must be followed exactly as listed here.

- **'H'** : hydrogen in water
- **'O'** : oxygen in water
- **'L'** : charge site on lone pair positions in an sp^3 hybridization of oxygen

Coulombic interactions

TIP5P uses atom-centered point charges to represent the electrostatic interactions. Here are the suggested charge assignments for water.

- **'H'**: 0.241
- **'O'**: 0.0
- **'L'**: -0.241

Improper torsions

There are no improper torsions for this force field.

Return to the Towhee Capabilities web page

Send comments to: Marcus G. Martin
Last updated: September 07, 2005

MCCCS Towhee (TraPPE Explict Hydrogen)

Overview

This section covers the TraPPE Explicit Hydrogen (TraPPE-EH) force field as it is implemented into the towhee_ff_TraPPE-EH file in the ForceFields directory. All of the Towhee atom types for the TraPPE Explicit Hydrogen force field are listed, along with a short description of their meanings. For more information about the TraPPE family of force fields see the Siepmann group web site. Note that TraPPE-EH is a Lennard-Jones (12-6) force field and can only be combined with other Lennard-Jones (12-6) force fields. It is particularily suitable for combination with the TraPPE United Atom force field. Any discrepencies (especially typos) from the published force field values are the sole responsibility of Marcus G. Martin, and I welcome feedback on how this implementation compares with other programs.

References for TraPPE-EH

TraPPE-EH is still under development by the Siepmann group. The current implementation includes only a subset of the total published parameters. Here is the current list of references.

- Potoff and Siepmann 2001

TraPPE-EH in Towhee

The official force field name for TraPPE Explicit Hydrogen in Towhee is 'TraPPE-EH'. Here I list all of the TraPPE-EH atom names for use in the towhee_input file, along with a brief description. I created these atom names in order to work with the molecule assembler. Please note that the capitalization and spacing pattern is important and must be followed exactly as listed here.

Carbon Dioxide
- **'C_co2'** : central carbon in carbon dioxide.
- **'O_co2'** : oxygen in carbon dioxide.

N_2
- **'N_n2'** : nitrogen atom in N_2.
- **'COM_n2'** : center of mass in N_2. This molecule is treated as a three site model with both of the nitrogen atoms bonded to the center-of-mass.

Coulombic interactions

TraPPE-EH uses atom-centered, and other interaction site centered, point charges to represent the electrostatic interactions. The 'bond increment' method for assigning charges is implemented for these molecules. Please see the inpstyle 2 documentation for further information on enabling this option. Here I list the bond increments for the atom names listed above. The value of the bond increment is added to the first atom listed and subtracted from the second atom listed.

- O_co2 - O_co2: -0.35
- N_n2 - COM_n2: -0.482

Improper torsions

TraPPE-EH does not utilize improper torsions.

Return to the Towhee Capabilities web page

Send comments to: Marcus G. Martin
Last updated: September 07, 2005

MCCCS Towhee (TraPPE United Atom)

Overview

This section covers the TraPPE United Atom (TraPPE-UA) force field as it is implemented into the towhee_ff_TraPPE-UA file in the ForceFields directory. All of the Towhee atom types for the TraPPE United Atom force field are listed, along with a short description of their meanings. For more information about the TraPPE United Atom force field see the Siepmann group web site. Note that TraPPE United Atom is a Lennard-Jones (12-6) force field and can only be combined with other Lennard-Jones (12-6) force fields. I would like to acknowledge Collin D. Wick, Jeff Potoff, and J. Ilja Siepmann for providing very useful guidance about implementing TraPPE-UA. Any discrepencies (especially typos) from the published TraPPE-UA force field values had better not exist considering I helped create this force field. The detail presented here for the TraPPE-UA force field is greater than the general level of detail for other forcefields due to the close relation between the authors of Towhee and TraPPE-UA. If someone does find a discrepancy with the published values I would be extremely grateful if they let me know about it. I welcome feedback on how this implementation compares with other programs.

References for TraPPE-UA

TraPPE-UA is still under development by the Siepmann group. Here is the current list of references.

- Martin and Siepmann 1998 (JPCB)
- Martin and Siepmann 1998 (TCA)
- Martin and Siepmann 1999
- Wick *et al.* 2000
- Chen *et al.* 2001
- Kamath *et al.* 2004
- Stubbs *et al.* 2004

Typos and comments for TraPPE-UA

Here I list torsions that I believe are missing in the published TraPPE-UA papers. These torsions are therefore not implemented into Towhee.

alkanes

- CH_x - CH - C - CH_y torsion is missing. Example: 2,3,3-trimethylbutane.
- CH_x - C - C - CH_y torsion is missing. Example: 2,2,3,3-quatramethylbutane.

alkenes

- CH_x - CH2(sp3) - C(sp2 double bonded to C) - CH_y torsion is missing. Example: 2-methyl-1-butene torsion involving only single bonds.
- CH_x - CH(sp3) - C(sp2 double bonded to C) - CH_y torsion is missing. Example: 2,3-dimethyl-1-butene torsion involving only single bonds.
- CH_x - C(sp3) - C(sp2 double bonded to C) - CH_y torsion is missing. Example: 2,3,3-trimethyl-1-butene torsion involving only single bonds.

alcohols

- CH_x - CH2 - CH - OH torsion is missing. Example: 1-methylethanol.
- CH_x - CH2 - C - OH torsion is missing. Example: 1,1-dimethylethanol.
- CH_x - CH2 - C - OH torsion is missing. Example: 1,1,1-trimethylethanol.

ketones

- CH_x - CH2(sp3) - C(sp2 double bonded to O) - CH_y torsion is missing. Example: buta-2-one torsion involving only single bonds.
- CH_x - CH(sp3) - C(sp2 double bonded to O) - CH_y torsion is missing. Example: 3-methylbuta-2-one torsion involving only single bonds.
- CH_x - C(sp3) - C(sp2 double bonded to O) - CH_y torsion is missing. Example: 3,3-dimethylbuta-2-one torsion involving only single bonds.

ethers
- CH_x - CH2(sp3) - CH(sp3) - O torsion is missing. Example: sec-butyl ethyl ether.
- CH_x - CH(sp3) - CH(sp3) - O torsion is missing. Example: 1,2-dimethylpropyl ethyl ether.
- CH_x - C(sp3) - CH(sp3) - O torsion is missing. Example: 1,2,2-trimethylpropyl ethyl ether.
- CH_x - CH2(sp3) - C(sp3) - O torsion is missing. Example: 1,1-dimethylpropyl ethyl ether.
- CH_x - CH(sp3) - C(sp3) - O torsion is missing. Example: 1,1,2-trimethylpropyl ethyl ether.
- CH_x - C(sp3) - C(sp3) - O torsion is missing. Example: 1,1,2,2-quatramethylpropyl ethyl ether.

A few angles for carboxylic acids were not listed in the Kamath *et al.* 2004 paper and here I list the missing angle terms and then also list the terms Jeff Potoff suggested for use with these angles.
- CH_2 - C(acid) - O in carboxylic acid is not listed. I used the similar CH_3 - C(acid) - O parameters.
- CH_2 - C(acid) = O in carboxylic acid is not listed. I used the similar CH_3 - C(acid) = O parameters.

The functional form for one of the torsions was listed incorrectly in Equation 5 of the Kamath *et al.* 2004 paper. That paper lists the torsion as

$$U_{torsion} = c_1[\ 1 + \cos(\ f + f_1\ )] + c_2\ [\ 1 - \cos(2\ f)]$$

but the actual torsion potential used in that work (according to personal communication with Jeff Potoff) is

$$U_{torsion} = c_1[\ 1 + \cos(\ f + f_1\ )] + c_2\ [\ 1 - \cos^2(f)]$$

and this correction was made in Towhee beginning with version 4.6.1.
A few torsions for carboxylic acids were not listed in the Kamath *et al.* 2004 paper and here I list the missing torsion terms and then also list the terms Jeff Pottof suggested for use with these torsions.
- CH_x - CH_2 - C(acid) = O in carboxylic acid is not listed. This uses the OCOH carboxylic torsion.
- CH_x - CH_2 - C(acid) - O in carboxylic acid is not listed. This uses the linear alkane CH_x - CH_2 - CH_2 - CH_x torsion.

The functional form used for alkenes is listed incorrectly in Equation 5 of the Wick *et al.* 2000 paper. That equation should read

$$u_{tors} = d_0\ (f - f_0)^2$$

TraPPE-UA in Towhee

The official force field name for TraPPE United Atom in Towhee is 'TraPPE-UA'. Here I list all of the TraPPE-UA atom names for use in the towhee_input file, along with a brief description. I created these atom names in order to work with the molecule assembler. TraPPE-UA is a united-atom force field which lumps hydrogens onto their neighboring carbons. Note that hydrogens that are bonded to atoms other than carbon are not lumped onto their neighboring atoms. The naming convention listed here began with version 4.4.1 as some changed were required in order to implement TraPPE-6. The general pattern of the names is a list of the atoms that make up the group, followed by the types of atoms that they are single bonded to, followed by the atoms they are double bonded to (prefixed with an '='), and ending in a code that represents the bonding pattern of the atom. The **Bond Name** is also listed to facilitate comparision with the bond increment method listed down in the coulombic section. Please note that the capitalization and spacing pattern is important and must be followed exactly as listed here.

Argon
- **'Ar'** : argon

Carbon (not bonded to any hydrogens)
- **'Ccccc(sp3)'** : an sp^3 carbon. This atom is single bonded to four carbon atoms. Bond name Csp3.
- **'Cccco(sp3)'** : an sp^3 carbon. This atom is single bonded to three carbon atoms and single bonded to one oxygen atom. Bond name Csp3.
- **'Ccc=c(sp2)'** : an sp^2 carbon. This atom is single bonded to two carbon atoms and double bonded to one carbon atom. Bond name Csp2.
- **'Ccc=o(sp2)'** : an sp^2 carbon. This atom is single bonded to two carbon atoms and double bonded to one oxygen atom. Bond name Ccc=o(sp2).
- **'Cco=o(sp2)'** : an sp^2 carbon. This atom is single bonded to one carbon atom, single bonded to one oxygen

atom, and double bonded to one oxygen atom. Bond name Cco=o(sp2).
- **'Caac(aro)'** : an aromatic carbon. This atom is bonded to two aromatic carbon atoms and single bonded to one carbon atom. Bond name Caro.
- **'Caaa(aro)'** : an aromatic carbon. This atom is bonded to three aromatic carbon atoms. Bond name Caro.

Carbon (united atom including 1 bonded hydrogen)
- **'CHccc(sp3)'** : united-atom consisting of an sp^3 carbon and one bonded hydrogen. This united-atom is single bonded to three carbon atoms. Bond name Csp3.
- **'CHcco(sp3)'** : united-atom consisting of an sp^3 carbon and one bonded hydrogen. This united-atom is single bonded to two carbon atoms and single bonded to one oxygen atom. Bond name Csp3.
- **'CHc=c(sp2)'** : united-atom consisting of an sp^2 carbon and one bonded hydrogen. This united-atom is single bonded to one carbon atom and double bonded to one carbon atom. Bond name Csp2.
- **'CHc=o(sp2)'** : united-atom consisting of an sp^2 carbon and one bonded hydrogen. This united-atom is single bonded to one carbon atom and double bonded to one oxygen atom. Bond name CHc=o(sp2).
- **'CHaa(aro)'** : united-atom consisting of an aromatic carbon and one bonded hydrogen. This united-atom is bonded to two aromatic carbon atoms. Bond name Caro.

Carbon (united atom including 2 bonded hydrogens)
- **'CH2**(sp3)'** : united-atom consisting of an sp^3 carbon and two bonded hydrogens. This united-atom is single bonded to any two non-hydrogen atoms. Bond name Csp3.
- **'CH2=*(sp2)'** : united-atom consisting of an sp^2 carbon and two bonded hydrogens. This united-atom is double bonded to any one non-hydrogen atom. Bond name Csp2.

Carbon (united atom including 3 bonded hydrogens)
- **'CH3*(sp3)'** : united-atom consisting of an sp^3 carbon and three bonded hydrogens. This united-atom is single bonded to any non-hydrogen atom. Bond name Csp3.

Carbon (united atom including 4 bonded hydrogens)
- **'CH4'** : methane

Helium
- **'He'** : helium

Hydrogen (not bonded to a carbon)
- **'Ho'** : hydrogen bonded to one oxygen. Bond name Ho.
- **'Hoacid'** : hydrogen bonded to one oxygen in a carboxylic acid. Bond name Hoacid.

Oxygen
- **'Och(sp3)'** : sp^3 oxygen atom that is single bonded to one carbon atom and single bonded to one hydrogen atom. Bond name Och(sp3).
- **'Occ(sp3)'** : sp^3 oxygen atom that is single bonded to two carbon atoms. Bond name Occ(sp3).
- **'O=c(sp2)'** : sp^2 oxygen atom that is double bonded to one carbon atom. Bond name O=c(sp2).

TraPPE-UA and the Molecule Assembler

The molecule builder is only partially functional for TraPPE-UA. The builder can assemble aliphatic hydrocarbons and alcohols, but it does not work for aromatics. In the case of alkenes, TraPPE-UA has different torsional potentials for cis and trans and these can be distinguished using the options in the 'advanced connectivity map'. TraPPE-UA also uses a rigid model for aromatics, which is currently problematic for Towhee.

Coulombic interactions

TraPPE-UA uses atom-centered point charges to represent the electrostatic interactions. Generally, there is no charge on any of the hydrocarbons, but there are exceptions once other atoms are mixed into the molecules. Note that there is a 0.5 scaled 1-4 coulombic interaction built into the most of the torsional potentials. This general rule is contradicted by some of the torsions for carboxylic acids. You can take a look at the TraPPE-UA force field literature to determine how to assign the charges, or you can try out the 'bond increment' method for assigning charges on these molecules. Here I list the bond increments using the bond names listed above. The value of the bond increment is added to the first atom listed and subtracted from the second atom listed.
- Csp3 - Csp3: 0.0

- Csp3 - Csp2: 0.0
- Csp3 - Caro: 0.0
- Csp2 - Csp2: 0.0
- Caro - Caro: 0.0
- Ho - Och(sp3): 0.435
- Csp3 - Och(sp3): 0.265
- Csp3 - Occ(sp3): 0.25
- CHc=o(sp2) - Csp3: 0.043
- CHc=o(sp2) - O=c(sp2): 0.482
- Ccc=o(sp2) - O=c(sp2): 0.424
- O=c(sp2) - Cco=o(sp2): -0.45
- Csp3 - Cco=o(sp2): 0.12
- Och(sp3) - Cco=o(sp2): -0.09
- Hoacid - Och(sp3): 0.37

Return to the Towhee Capabilities web page

Send comments to: Marcus G. Martin

Last updated: January 09, 2007

MCCCS Towhee (TraPPE United Atom Flexible Bonds)

Overview

This section covers the TraPPE United Atom Flexible Bonds (TraPPE-UAf) force field as it is implemented into the towhee_ff_TraPPE-UAf file in the ForceFields directory. All of the Towhee atom types for this force field are listed, along with a short description of their meanings. For more information about the TraPPE force fields see the Siepmann group web site. Note that TraPPE United Atom Flexible bonds is a Lennard-Jones (12-6) force field and can only be combined with other Lennard-Jones (12-6) force fields. This forcefield is a subset of the rigid bond TraPPE-UA force field with the change that the bond lengths are no longer rigid and are instead flexible using parameters from some other papers. Any discrepencies (especially typos) from the published TraPPE-UA force field values had better not exist considering I helped create this force field. If someone does find a discrepancy with the published values I would be extremely grateful if they let me know about it. I welcome feedback on how this implementation compares with other programs.

References for TraPPE-UAf

TraPPE-UAf is still under development by the Siepmann group. The alkane parameters are taken from the first two TraPPE papers, and the flexible bond lengths are from Mundy *et al.*.

- Mundy *et al.* 1996
- Martin and Siepmann 1998 (JPCB)
- Martin and Siepmann 1998 (TCA)

Typos and comments for TraPPE-UAf

Please see the closely related TraPPE-UA force field for typos and comments about this force field.

TraPPE-UAf in Towhee

The official force field name for TraPPE United Atom Flexible Bonds in Towhee is 'TraPPE-UAf'. Here I list all of the TraPPE-UAf atom names for use in the towhee_input file, along with a brief description. These atom name are the same as those in the rigid bond TraPPE-UA force field and have the same meanings. Please note that the capitalization and spacing pattern is important and must be followed exactly as listed here.

Carbon (not bonded to any hydrogens)

- **'Ccccc(sp3)'** : an sp^3 carbon. This atom is single bonded to four carbon atoms. Bond name Csp3.

Carbon (united atom including 1 bonded hydrogen)

- **'CHccc(sp3)'** : united-atom consisting of an sp^3 carbon and one bonded hydrogen. This united-atom is single bonded to three carbon atoms. Bond name Csp3.

Carbon (united atom including 2 bonded hydrogens)

- **'CH2**(sp3)'** : united-atom consisting of an sp^3 carbon and two bonded hydrogens. This united-atom is single bonded to any two non-hydrogen atoms. Bond name Csp3.

Carbon (united atom including 3 bonded hydrogens)

- **'CH3*(sp3)'** : united-atom consisting of an sp^3 carbon and three bonded hydrogens. This united-atom is single bonded to any non-hydrogen atom. Bond name Csp3.

Carbon (united atom including 4 bonded hydrogens)

- **'CH4'** : methane

Coulombic interactions

TraPPE-UA uses atom-centered point charges to represent the electrostatic interactions. There are no coulombic interactions for the atom types currently implemented in this force field. Here I list the bond increments using the bond names listed above. The value of the bond increment is added to the first atom listed and subtracted from the second atom listed.

- Csp3 - Csp3: 0.0

Improper torsions

TraPPE-UAf does not utilize improper torsions.

Return to the Towhee Capabilities web page

Send comments to: Marcus G. Martin
Last updated: November 12, 2005

MCCCS Towhee (Walther *et al.* 2001)

Overview

This section covers the Walther *et al.* (Walt2001) force field as it is implemented into the towhee_ff_Walt2001 file in the ForceFields directory. This force field was originally designed for fluid simulations of water and carbon nanotubes. Note that this is a Lennard-Jones force field that uses explicit mixing rules so it is not easily combined with other force fields. Any discrepencies (especially typos) from the published values are the sole responsibility of Marcus G. Martin, and I welcome feedback on how this implementation compares with other programs.

References for Walt2001

What I term Walt2001 is described in the following paper.

- Walther *et al.* 2001

Typos and comments for Walt2001

There are some typos in the Walt2001 paper. Here I list places where my implementation does not completely agree with what is written in the Walt2001 paper.

- Table 1 of Walther *et al.* 2001 lists no values for the angle force constant and two for the torsional force constant. This implementation assumes that the second of these values (25.12 kJ mol^{-1}) is the torsional force constant, while the first value (562.2 kJ mol^{-1}) is the angle force constant.
- Table 2 of Walther *et al.* 2001 lists a charge of -0.41 e for the hydrogen atom in water. This implemenation assumes they really meant a charge of +0.41 e for the hydrogen atom in water.

Walt2001 in Towhee

The official force field name for Walther *et al.* 2001 in Towhee is 'Walt2001'. Here I list all of the atom names for use in the towhee_input file, along with a brief description. I created these atom names in order to work with the molecule assembler. Please note that the capitalization and spacing pattern is important and must be followed exactly as listed here.

- **'C'** : Carbon in a carbon nanotube.
- **'O'** : Oxygen in water.
- **'H'** : Hydrogen in water.

Coulombic interactions

This force field assigns coulombic interactions to the water molecules. In the original paper they also used quadrupole moments on the carbon atoms, but they found these had no significant effect on the results and so the quadrupole moment is not included in this implementation. The 'bond increment' method of **charge_assignment** is implemented for this forcefield. Please see the inpstyle 2 documentation for further information on enabling this option. Otherwise, you are welcome to manually set the charges. Here I list the charges transfered between atom pairs as assigned by the 'bond increment' method.

- O - H: -0.41
- C - C: 0.0

Return to the Towhee Capabilities web page

Send comments to: Marcus G. Martin
Last updated: September 07, 2005

MCCCS Towhee (Elliott 2002)

Overview

This section covers Elliott 2002 parameters as they are implemented into the towhee_ff_Elli2002 file in the ForceFields directory. This force field contains parameters for some linear alkanes and benzene and uses the 'Multiwell' force field. It is designed for use with the Lorentz-Berthelot mixing rule. I would like to acknowledge J. Richard Elliott for providing useful guidance about implementing this force field. Any discrepencies (especially types) from the published Elli2002 force field values are the sole responsibility of Marcus G. Martin, and I welcome feedback on how this implementation compares with other programs.

References for Elliott 2002

- Elliott 2002

Elliott 2002 in Towhee

The official force field name for Elliott 2002 in Towhee is 'Elli2002'. Here is a list of all atom names currently in use for the towhee_input file, along with a brief description. Please note that the capitalization and spacing pattern is important and must be followed exactly as listed here.

- **'methane'** : united-atom methane
- **'ethane'** : united-atom CH_3 in ethane
- **'butane-CH3'** : united-atom CH_3 in butane
- **'butane-CH2'** : united-atom CH_2 in butane
- **'hexane-CH3'** : united-atom CH_3 in hexane
- **'hexane-CH2'** : united-atom CH_2 in hexane
- **'octane-CH3'** : united-atom CH_3 in octane
- **'octane-CH2'** : united-atom CH_2 in octane
- **'benzene'** : united-atom CH_2 in benzene

Coulombic interactions

This force field does not utilize coulombic interactions.

Return to the Towhee Capabilities web page

Send comments to: Marcus G. Martin
Last updated: December 03, 2005

MCCCS Towhee (Ding-Andersen germanium)

Overview

This section covers the Ding-Andersen germanium parameters as they are implemented into the towhee_ff_Ding1986 file in the ForceFields directory. All of the Towhee atom types for this force field are listed, along with a short description of their meanings. This force field was intended for solid germanium. It uses the Stillinger-Weber potential type. Any discrepencies (especially typos) from the published values are the sole responsibility of Marcus G. Martin, and I welcome feedback on how this implementation compares with other programs.

References for Ding-Andersen germanium

- Ding and Andersen 1986

Vink *et al.* in Towhee

The official force field name for Ding-Andersen germanium parameters is 'Ding1986'. Here I list the atom names for use in the towhee_input file, along with a brief description (although the meaning of the atom names should be obvious). Please note that the capitalization and spacing pattern is important and must be followed exactly as listed here.

  - **'Ge'** : germanium

Coulombic interactions

There are generally no coulombic interactions for this potential.

Improper torsions

This force field does not have improper torsions.

Proteins

This force field does not have parameters for proteins.

Return to the Towhee Capabilities web page

Send comments to: Marcus G. Martin
Last updated: September 07, 2005

MCCCS Towhee (Stillinger-Weber Silicon)

Overview

This section covers the Stillinger-Weber Silicon parameters as they are implemented into the towhee_ff_Stil1985 file in the ForceFields directory. All of the Towhee atom types for the Stillinger-Weber silicon force field are listed, along with a short description of their meanings. This force field was intended for solid silicon. Any discrepencies (especially typos) from the published values are the sole responsibility of Marcus G. Martin, and I welcome feedback on how this implementation compares with other programs.

References for Stillinger-Weber silicon

- Stillinger and Weber 1985

Stillinger-Weber in Towhee

The official force field name for Stillinger-Weber silicon parameters is 'Stil1985'. Here I list the atom names for use in the towhee_input file, along with a brief description (although the meaning of the atom names should be obvious). Please note that the capitalization and spacing pattern is important and must be followed exactly as listed here.

  - **'Si'** : silicon

Coulombic interactions

There are generally no coulombic interactions for this potential.

Improper torsions

This force field does not have improper torsions.

Proteins

This force field does not have parameters for proteins.

Return to the Towhee Capabilities web page

Send comments to: Marcus G. Martin
Last updated: September 07, 2005

MCCCS Towhee (Vink *et al.* silicon)

Overview

This section covers the Vink *et al.* silicon parameters as they are implemented into the towhee_ff_Vink2001 file in the ForceFields directory. All of the Towhee atom types for this force field are listed, along with a short description of their meanings. This force field was intended for solid silicon. Any discrepencies (especially typos) from the published values are the sole responsibility of Marcus G. Martin, and I welcome feedback on how this implementation compares with other programs.

References for Vink *et al.* silicon

- Vink *et al.* 2001

Vink *et al.* in Towhee

The official force field name for Vink *et al.* silicon parameters is 'Vink2001'. Here I list the atom names for use in the towhee_input file, along with a brief description (although the meaning of the atom names should be obvious). Please note that the capitalization and spacing pattern is important and must be followed exactly as listed here.

  - **'Si'** : silicon

Coulombic interactions

There are generally no coulombic interactions for this potential.

Improper torsions

This force field does not have improper torsions.

Proteins

This force field does not have parameters for proteins.

Return to the Towhee Capabilities web page

Send comments to: Marcus G. Martin
Last updated: September 07, 2005

MCCCS Towhee (Square Well)

Overview

This section covers Square Well parameters as they are implemented into the towhee_ff_SquareWell file in the ForceFields directory. It is a bit strong to call this a force field, as all there is not much to a square well potential, but this file explains how to use the builder with the square well potential, and how to add in any other square well parameters that you might want to use in a simulation.

Square Well in Towhee

The official force field name for Square Well in Towhee is 'SquareWell'. Here I list all of the Square Well atom names currently in use for the towhee_input file, along with a brief description (although the meaning of the atom names should be obvious). If you want to add any new parameters into Towhee just take a look in the towhee_input classical_potential documentation for a complete description of the functional form. I am happy to add some more of these parameter sets into the release version as well (only takes a few minutes). The distance units are in Angstroms, and the well depth is in Kelvin divided by Boltzmann's constant. Please note that the capitalization and spacing pattern is important and must be followed exactly as listed here.

- **'type_a'** : sigma [nbcoeff(1)] is 1.0, lamda [nbcoeff(2)] is 1.5, and the well depth [nbcoeff(3)] is 1.0.
- **'type_b'** : sigma [nbcoeff(1)] is 2.0, lamda [nbcoeff(2)] is 3.0, and the well depth [nbcoeff(3)] is 1.0.

Coulombic interactions

It is possible to combine the square well potential with point charges. Simply assign the point charges to the atoms as you see fit.

Return to the Towhee Capabilities web page

Send comments to: Marcus G. Martin
Last updated: September 07, 2005

MCCCS Towhee (Vega *et al.* 1992)

Overview

This section covers Vega *et al.* 1992 parameters as they are implemented into the towhee_ff_Vega1992 file in the ForceFields directory. This force field is simply a collection of Square Well parameters that Vega *et al.* used in a vapor-liquid coexistence curve study. Suitable bonded parameters for this force field have also been added into this implementation.

References for Vega 1992

- Vega *et al.* 1992

Vega 1992 in Towhee

The official force field name for Vega *et al.* 1992 in Towhee is 'Vega1992'. Here is a list of all Vega1992 atom names currently in use for the towhee_input file, along with a brief description (although the meaning of the atom names should be obvious). Please note that the capitalization and spacing pattern is important and must be followed exactly as listed here.

- **'lam_1.25'** : sigma [nbcoeff(1)] is 1.0, lamda [nbcoeff(2)] is 1.25, and the well depth [nbcoeff(3)] is 1.0.
- **'lam_1.375'** : sigma [nbcoeff(1)] is 1.0, lamda [nbcoeff(2)] is 1.375, and the well depth [nbcoeff(3)] is 1.0.
- **'lam_1.5'** : sigma [nbcoeff(1)] is 1.0, lamda [nbcoeff(2)] is 1.5, and the well depth [nbcoeff(3)] is 1.0.
- **'lam_1.75'** : sigma [nbcoeff(1)] is 1.0, lamda [nbcoeff(2)] is 1.75, and the well depth [nbcoeff(3)] is 1.0.
- **'lam_2.0'** : sigma [nbcoeff(1)] is 1.0, lamda [nbcoeff(2)] is 2.0, and the well depth [nbcoeff(3)] is 1.0.

Coulombic interactions

It is possible to combine the square well potential with point charges. Simply assign the point charges to the atoms as you see fit.

Return to the Towhee Capabilities web page

Send comments to: Marcus G. Martin
Last updated: September 07, 2005

MCCCS Towhee (UFF)

Overview

This section covers the Universal Force Field (UFF) as it is implemented into the towhee_ff_UFF file in the ForceFields directory. All of the Towhee atom types for this force field are listed, along with a short description of their meanings. Note that UFF is a Lennard-Jones style force field, but has some special additional parameters and so cannot be directly combined with other force fields. You need to use the classical_potential 'UFF 12-6' for the UFF force field and the suggested mixing rules are 'Geometric'. Please note that the UFF paper contains a method to generate Exponential-6 potentials from their data set as well as the 12-6. If anyone is interested in an Exponential-6 version of this force field please let me know and I'll be happy to implement that as well. I would like to acknowledge Anthony Rappe for kindly answering my questions about this force field. Any discrepencies (especially typos) from the published force field values are the sole responsibility of Marcus G. Martin, and I welcome feedback on how this implementation compares with other programs.

References for UFF

Most of the parameters for UFF are published in Table~1 of the primary reference for UFF.

- Rappe *et al.* 1992

However, that paper refers to another paper (their reference 13) as submitted, that unfortunately appears never to have been published. The "GMP electronegativies" (*X*) in that reference are required in order to compute the bonded force constants and equilibrium distances. A partial list of the GMP Electronegativies is available in a related paper.

- Rappe and Goddard 1991

I also managed to get ahold of the unpublished parameter files for UFF and used this to fill in the rest of the missing elements.

Typos and comments for UFF

There are some obvious typos in the UFF paper, and I believe there are a few subtle ones as well. Here I list places where my implementation does not completely agree with what is written in the UFF paper.

- Equation (2) of Rappe *et al.* 1992 is written as follows.
  $r_{IJ} = r_I + r_J + r_{BO} + r_{EN}$
  However, this method does not result in agreement with their published equilibrium bond lengths. Anthony Rappe informed me that this equation is in error and I have instead implemented the following (beginning with Version 4.4.2).
  $r_{IJ} = r_I + r_J + r_{BO} - r_{EN}$
- Equation (13) of Rappe *et al.* 1992 is written with some mistakes in the superscripts and subscripts. Here is the equation as implemented into Towhee (beginning with Version 4.4.2).
  $K_{IJK} = \text{beta} (Z_I^* Z_K^* / r_{IK}^5) r_{IJ} r_{JK} [3 r_{IJ} r_{JK} (1 - \text{Cos}^2(\text{theta}_0)) - r_{IK}^2 \text{Cos}(\text{theta}_0)]$
- The final sentence before Equation (19) of Rappe *et al.* 1992 suggests that the two different methods for determining the inversion angle differ by a factor of Pi. These two methods actually differ by a factor of Pi/2.
- Lawrencium is named **Lw6+3** in Rappe *et al.* 1992, but this is not consistent with the accepted abrieviation for that element. Therefore this element is listed as **Lr6+3** in Towhee.
- The paragraph following Equation 17 in Rappe *et al.* 1992 is confusing because it refers to default values for the "first through sixth periods", but then only lists five values. Originally, I assigned these 5 values to the first through fifth periods, but after

discussions with Jon Baker I now believe these values are appropriate for the second through sixth periods. Begining with verion 4.7.11 the default values for nbcoeff(12) are as follows.
  - Period 2 (Li through Ne): 2.0
  - Period 3 (Na through Ar): 1.25
  - Period 4 (K through Kr): 0.7
  - Period 5 (Rb through Xe): 0.2
  - Period 6 (Cs through Rn): 0.1
- Equation 10 and the preceding text in Rappe *et al.* 1992 does not accurately reflect the implementation of bending angles in UFF. For the linear case the equation should actually read
  $U = K_{IJK}/n^2$ [1 + Cos(n theta)]
  In addition, this equation is used for tetrahedral cases (3rd character is '3') when the equilibrium angle is 90.0. It is correct as written for the other cases. This change was made to Towhee starting with version 4.11.0. Previously the Equation 10 was used as written. Thanks to Jon Baker for identifying this problem.

There are a handful of final parameters listed in Rappe *et al.* 1992 that allow a comparison of the Towhee implementation with their work.

- Towhee has an equilibrium **C_R - N_R** bond length of 1.3568 Å in good agreement with the statement on page 10026 of Rappe *et al.* 1992 that their bond length agrees well with 1.366 Å)
- Towhee has a **C_R - N_R** force constant of 325378.3 K = 646.59 kcal/mol that is half of the force constant of 1293 kcal/mol on page 10027 of Rappe *et al.* 1992. In the same sentence they reference the Weiner1986 force field and claim it has a force constant of 980 kcal/mol*Å^2. The table in the appendix of Weiner *et al.* 1986. states a **C-N** force constant of 490 kcal/mol*Å^2 and Equation 1 of that same paper uses a harmonic potential of form $K_R(R - R_0)^2$. It appears that the UFF authors doubled all of the force constants in this sentence, perhaps to bring the force constants into agreement with the frequently used 1/2 $K(R - R_0)^2$ form of the harmonic potential.
- Towhee has a **C_3 - N_R - C_R** force constant of 106165.606 K = 210.97397 kcal/mol rad^2 that is almost exactly twice the force constant of 105.5 kcal/mol*rad^2 stated on page 10028 of Rappe *et al.* 1992. In the same sentence they reference the Weiner1986 force field and claim it has a force constant of 100 kcal/mol*rad^2. The table in the appendix of Weiner *et al.* 1986. states a **C-N-CT** force constant of 50 kcal/mol*rad^2 and Equation 1 of that same paper uses a harmonic angle potential of form $K_\theta(\theta - \theta_0)^2$. Perhaps the UFF authors accidentally halved their reported force constant instead of doubling it for comparison with the other force fields.

UFF in Towhee

The official force field name for UFF in Towhee is 'UFF'. Here I list all of the atom names for use in the towhee_input file, along with a brief description taken from the UFF literature. UFF uses a five-character label to describe every element. The first two letters are the chemical symbol (appended with an underscore for single letter elements). The third character describes the geometry of the molecule as follows.
- 1: linear
- 2: trigonal
- R: resonant
- 3: tetrahedral

- 4: square planar
- 5: trigonal bipyramidal
- 6: octahedral

The fourth and fifth characters are there to help distinguish between otherwise similar atoms (for example, the charge state of metals and special characters for certain hydrogen and oxygen atoms). Towhee follows the UFF naming convension exactly, except for Lawrencium where the correct 'Lr' abreviation is used instead of the 'Lw' in the original paper. The element names are generally obvious (given the rules above), but a notes are added to some potentially confusing elements. Please note that the capitalization and spacing pattern is important and must be followed exactly as listed here.

- **'H_'**
- **'H_b'**: hydrogen bridging between two boron atoms
- **He4+4**: helium
- **'Li'**
- **'Be3+2'**
- **'B_3'**
- **'B_2'**
- **'C_3'**
- **'C_R'**
- **'C_2'**
- **'C_1'**
- **'N_3'**
- **'N_R'**
- **'N_2'**
- **'N_1'**
- **'O_3'**
- **'O_3_z'**: oxygen in a zeolite framework
- **'O_R'**
- **'O_2'**
- **'O_1'**
- **'F_'**
- **'Ne4+4'**
- **'Na'**
- **'Mg3+2'**
- **'Al3'**
- **'Si3'**
- **'P_3+3'**
- **'P_3+5'**
- **'P_3+q'**
- **'S_3+2'**
- **'S_3+4'**
- **'S_3+6'**
- **'S_R'**
- **'S_2'**
- **'Cl'**
- **'Ar4+4'**
- **'K_'**
- **'Ca6+2'**
- **'Sc3+3'**
- **'Ti3+4'**
- **'V_3+5'**
- **'Cr6+3'**
- **'Mn6+2'**
- **'Fe3+2'**

- **'Fe6+2'**
- **'Co6+3'**
- **'Ni4+2'**
- **'Cu3+1'**
- **'Zn3+2'**
- **'Ga3+3'**
- **'Ge3'**
- **'As3+3'**
- **'Se3+2'**
- **'Br'**
- **'Kr4+4'**
- **'Rb'**
- **'Sr6+2'**
- **'Y_3+3'**
- **'Zr3+4'**
- **'Nb3+5'**
- **'Mo6+6'**
- **'Mo3+6'**
- **'Tc6+5'**
- **'Ru6+2'**
- **'Rh6+3'**
- **'Pd4+2'**
- **'Ag1+1'**
- **'Cd3+2'**
- **'In3+3'**
- **'Sn3'**
- **'Sb3+3'**
- **'Te3+2'**
- **'I_'**
- **'Xe4+4'**
- **'Cs'**
- **'Ba6+2'**
- **'La3+3'**
- **'Ce6+3'**
- **'Pr6+3'**
- **'Nd6+3'**
- **'Pm6+3'**
- **'Sm6+3'**
- **'Eu6+3'**
- **'Gd6+3'**
- **'Tb6+3'**
- **'Dy6+3'**
- **'Ho6+3'**
- **'Er6+3'**
- **'Tm6+3'**
- **'Yb6+3'**
- **'Lu6+3'**
- **'Hf3+4'**
- **'Ta3+5'**
- **'W_6+6'**
- **'W_3+4'**
- **'W_3+6'**
- **'Re6+5'**
- **'Re3+7'**

- **'Os6+6'**
- **'Ir6+3'**
- **'Pt4+2'**
- **'Au4+3'**
- **'Hg1+2'**
- **'Tl3+3'**
- **'Pb3'**
- **'Bi3+3'**
- **'Po3+2'**
- **'At'**
- **'Rn4+4'**
- **'Fr'**
- **'Ra6+2'**
- **'Ac6+3'**
- **'Th6+4'**
- **'Pa6+4'**
- **'U_6+4'**
- **'Np6+4'**
- **'Pu6+4'**
- **'Am6+4'**
- **'Cm6+3'**
- **'Bk6+3'**
- **'Cf6+3'**
- **'Es6+3'**
- **'Fm6+3'**
- **'Md6+3'**
- **'No6+3'**
- **'Lr6+3'**

Coulombic interactions

The UFF parameters were derived without the use of point charges on the atoms and I believe the consensus of the original authors is to use this force field without any additional charges. One notable proponent of not using partial charges for UFF is A.K. Rappe who states this quite strongly in his UFF FAQ. If you feel an overwhelming desire to assign partial charges then that is allowed in Towhee, but there is no *official* reference for UFF with partial charges. However, if you were so inclined then the QEq method of Rappe and Goddard (1991) might be appropriate.

Improper torsions

UFF uses an improper torsion (called an inversion in their paper) on any atom (I) that is bonded to exactly three other atoms (J,K, and L). The improper considers the angle each of the vectors (IJ, IK or IL) makes with a plane described by the other substituants. For example, the angle between the IJ vector and the IKL plane. Towhee options currently require the user to specify all improper torsions, but you may toggle the type to 0 to allow Towhee to automatically determine the appropriate parameters for each improper torsion.

Return to the Towhee Capabilities web page

Send comments to: Marcus G. Martin
Last updated: October 19, 2005

MCCCS Towhee (Kramer-Farragher-van Beest-van Santen)

Overview

This section covers the Kramer-Farragher-van Beest-van Santen (KFvBvS) force field as it is implemented into the towhee_ff_KFvBvS file in the ForceFields directory. All of the Towhee atom types for the KFvBvS force field are listed, along with a short description of their meanings. The KFvBvS force field uses the exponential-6 potential type and therefore must also use explicit combining rules. It therefore cannot be combined with other force fields. Any discrepencies (especially typos) from the published force field values are the sole responsibility of Marcus G. Martin, and I welcome feedback on how this implementation compares with other programs.

References for KFvBvS

This force field comes from the work of Kramer, Farragher, van Beest, and van Santen. I created the name for this force field from the initials of the last names of the authors. The parameters were taken from the mixed SCF empirical force field listed in

- van Beest *et al.* 1990
- Kramer *et al.* 1991

KFvBvS in Towhee

The official force field name for these parameters is either 'KFvBvS'. Here I list all of the KFvBvS atom names for use in the towhee_input file, along with a brief description of what atoms we are talking about. This force field is intended for use with silacious materials (zeolites, silica, etc.). For some reason they did not include repulsive potentials between many of the atoms and Cl. Thus, Cl can only be used in combination with other negative ions, or with Na. Please note that the capitalization and spacing pattern is important and must be followed exactly as listed here.

- **'Al'** : aluminum
- **'Cl'** : chlorine
- **'Na'** : sodium
- **'O'** : oxygen
- **'P'** : phosphorous
- **'Si'** : silicon

Coulombic interactions

Atom centered point charges are used to represent the electrostatic interactions. Below is a list of charges taken from the mixed SCF empirical force field in Kramer *et al.* 1991

- **'Al'** : 1.4
- **'Cl'** : -1.0
- **'Na'** : 1.0
- **'O'** : -1.2
- **'P'** : 3.4
- **'Si'** : 2.4

Improper torsions

There are no improper torsions in this force field.

Proteins

The KFvBvS force field is designed for zeolites and is not suited for proteins.

Return to the Towhee Capabilities web page

Send comments to: Marcus G. Martin
Last updated: September 07, 2005

MCCCS Towhee (MGM Stereochemistry Enforcer)

Overview

This section covers the MGM Stereochemistry Enforcer improper torsion as implemented into the towhee_ff_mgmstereo file in the ForceFields directory. This is by no means a complete force field, it only exists to enforce L or D (or if you prefer R or S) stereochemistry. This is needed because configurational-bias is easily able to switch stereochemistry of a center during a configuartional-bias regrowth. This is remedied by added this improper torsion onto the stereochemical center.

MGM Stereochemistry Enforcer in Towhee

There are no atom names for this force field, it is used in combination with another force field.

Coulombic interactions

There are no specific coulombic interactions for this force field as it is used with other force fields.

Improper torsions

The 'mgmstereo' improper torsion can be combined with other forcefields by listing an improper torsion with a type of -1. This triggers a special flag in assemble.F to look for the mgmstereo improper torsion instead of searching the atom types of the actual force field for the molecule. In order to use this improper you need to define three improper torsions. If you wanted to define an "L" stereocenter on an amino acid C_{alpha} you would define the following torsions from the C_{alpha} atom. You would list the actual atom number instead of the names, but this gives the idea.
N H_{alpha} C_{beta} -1
N C_{beta} C(=O) -1
N C(=O) H_{alpha} -1
In order to define a "D" stereocenter you simply reverse the atom orders.
N C_{beta} H_{alpha} -1
N C(=O) C_{beta} -1
N H_{alpha} C(=O) -1

Proteins

This is often used with a protein force field. See the improper torsion section for more information.

Return to the Towhee Capabilities web page

Send comments to: Marcus G. Martin
Last updated: September 07, 2005

MCCCS Towhee input Version 6.0.x

MCCCS Towhee (towhee_input Version 6.0.x)

Overview

This section covers the variables that are set in the towhee_input file Version 6.0.x. Each variable is listed along with its type (logical, character, integer, or double precision). towhee_input is the main input file for Towhee and is generally the only file that needs to be edited on a regular basis. It has a regimented style to the input. The variables are described here in the order they appear in this file. Please look at one of the example files (available with the code package) for the precise file format.

Note that for each variable listed below you must include the name of the variable on the previous line. In addition, the variables that are subsets of various Monte Carlo moves must be indented 10 spaces.

All Towhee parameter files (including towhee_input) allow internal comments for user convenience. Such comments are lines which begin with the '#' character; the entire line is then ignored.

Some variables are optional. If the variable name is not included in towhee_input it will be set to the specified default value. Optional input file values are marked as *optional parameter*.

Bug reports and feature enhancements for 6.0.x versions

- 6.0.0: Added the number of molecule types into the towhee_initial/towhee_final files in order to assist in other codes parsing those files. Added the 'Hard Ball' **fieldtype**. Added the optional **controlstyle** variable for an easy, automatic method for setting many of the output and update parameters based upon the value of **nstep**.

towhee_input file differences from version 5.3.x

- The 'Hard Ball' field was added as an additional option for the **fieldtype** variable.
- The optional parameter **controlstyle** was added for a convenient way to set many of the output and update frequencies to suggested values for either equilibration or production work.

Quick links

Links to other towhee_input information pages, described elsewhere on this page:

- Nonbonded potential types are described in the classical_potential section.
- Information used to describe molecule-specific forcefield information is in the inpstyle section.
- towhee_input database format information.
- Various constants declared at compile-time found in preproc.h.

Variable explanations for towhee_input

inputformat (character string)
- 'Towhee' : reads in the input variables following the format for Towhee. This format is described in this file.
- 'LAMMPS' : reads in the input variables from the lammps_input and lammps_data files. Outputs files suitable for use with Towhee.
- 'Database' : reads in the input variables from the database_input file. Runs energy calculations for a database of conformations. See the towhee_input database format for more information about this feature.

randomseed (integer)
- The 32 bit integer seed that is used to initialize the ranlux random number generator. Must be positive.

random_luxlevel (integer)

An integer value that defines the luxury level used by the random number generator. See James 1994 for more information about the luxury level.
- 0: Does not discard any pseudorandom numbers from the sequence.
- 1: Periodically discards 24 pseudorandom numbers from the sequence.

- 2: Periodically discards 73 pseudorandom numbers from the sequence.
- 3: Periodically discards 199 pseudorandom numbers from the sequence. This is the suggested value for this variable.
- 4: Periodically discards 365 pseudorandom numbers from the sequence.

random_allow_restart (logical)

- .TRUE. if you want to restart the random number generator from the sequence of integers in towhee_initial if possible. This is done when the simulation is restarting from a towhee_initial file generated by Version 5.0.0 or later. If this is not possible then Towhee will initialize the random number generator using the **randomseed**.
- .FALSE. if you always want to initialize the random number generator using the **randomseed**.

ensemble (character string of size 30)

- 'npt' : Isobaric-Isothermal Ensemble. The volume moves for each simulation box are performed in an exchange with an external pressure bath set at a specified pressure. The total number of molecules and the temperature are conserved. See McDonald 1972 for the initial Monte Carlo molecular simulation work using the isobaric-isothermal ensemble. Combining this ensemble with any of the two-box molecule transfer moves turns it into the NPT-Gibbs ensemble. See Panagiotopoulos 1987, Panagiotopoulos *et al.* 1988, and Smit *et al.* 1989 for the initial Monte Carlo molecular simulation work using the Gibbs ensemble.
- 'nvt' : Canonical Ensemble. The total volume of the system, The total number of each type of molecule in the system, and the temperature are conserved. This is the prototypical ensemble for Monte Carlo molecular simulation and was first used in that context by Metropolis *et al.* 1953. Combining this ensemble with any of the two-box molecule transfer moves turns it into the NVT-Gibbs ensemble. See Panagiotopoulos 1987, Panagiotopoulos *et al.* 1988, and Smit *et al.* 1989 for the initial Monte Carlo molecular simulation work using the Gibbs ensemble.
- 'uvt' : Grand Canonical Ensemble. The total volume of the system is conserved. The total number of molecules in the system equilibrates with an external ideal gas bath set at a specified chemical potential.
- 'pseudo-bubble' : *pseudo*-bubble point ensemble. The total volume of the system is conserved. The total number of each type of molecule in one of the boxes is conserved. The total number of each type of molecule in the other simulation boxes is adjusted in order to reproduce the chemical potential of the molecules in the conserved box.

temperature (double precision)

- The temperature in Kelvin.

The variable in this subsection is only included in the input file if **ensemble** is set to 'npt'

pressure (double precision)

- The external pressure in kPa.

End of the subsection only included if **ensemble** is 'npt'

The variable in this subsection is only included in the input file if **ensemble** is set to 'uvt'

tmmc_flag (logical) *optional parameter*

- .TRUE. if you want to perform transition matrix Monte Carlo (TMMC) on a single component system in the grand canonical ensemble. This will require other variables to be specified below.
- .FALSE. if you do not want to perform transition matrix Monte Carlo (TMMC). This is the default value if the parameter is not specified.

End of the subsection only included if **ensemble** is 'uvt'

---

nmolty (integer)

- The total number of molecule types in the simulation. This must be less then or equal to NTMAX.

nmolectyp (integer) [one value for each molecule type]

- The number of molecules of each type (listed sequentially on a single line). For the constant N ensembles (nvt, npt) this is the actual number of molecules in the simulation. For the constant chemical potential ensembles (uvt) this is the maximum number of molecules allowed in the simulation.

The variable in this subsection is only included in the input file if **ensemble** is set to 'uvt'

chempot (double precision)

- The real chemical potential (this includes intramolecular portions and is identical to the CB chemical potential output by the code) for molecules of each type (listed sequentially on a single line). The units are in Kelvin (identical to the output CB chemical potential).

End of the subsection only included if **ensemble** is 'uvt'

---

numboxes (integer)

- The number of simulation boxes in the system. This value must be less than or equal to MAXBOX Note that many of the variables below depend upon **numboxes** as information is required based on the number of simulation boxes (such as box lengths) and some Monte Carlo moves are only valid for multiple box ensembles.

stepstyle (character string of length 10)

- 'cycles': Run a Monte Carlo simulation for **nstep** Monte Carlo cycles. A cycle is equal to **N** Monte Carlo moves, where **N** is the number of molecules in the system.
- 'moves': Run a Monte Carlo simulation for **nstep** Monte Carlo moves.

nstep (integer)

- The number of Monte Carlo steps to perform where each step is either a full Monte Carlo cycles (if **stepstyle** is 'cycles') or a single move (if **stepstyle** is 'moves')

controlstyle (character*20) *optional parameter*

- 'equilibration': designed for use when you are first setting up a simulation and still need to rapidly equilibrate all of the maximum update values. This option sets the following parameters based upon the value of **nstep**.
  - **printfreq** = **nstep**/10
  - **blocksize** = **nstep**/5
  - **moviefreq** = **nstep**/5
  - **backupfreq** = **nstep**/5
  - **runoutput** = 'full'
  - **pdb_output_freq** = **nstep**/5
  - **loutdft** = .FALSE.
  - **loutlammps** = .FALSE.
  - **loutdlpoly** = .FALSE.
  - **louthist** = .FALSE.
  - **pressurefreq** = 10, unless **ensemble** = 'uVT' in which case **pressurefreq** = 0
  - **trmaxdispfreq** = **nstep**/100
  - **volmaxdispfreq** = **nstep**/100
- 'production': designed for use during production runs. This option sets the following parameters based upon the value of **nstep**.
  - **printfreq** = **nstep**/10
  - **blocksize** = **nstep**/5
  - **moviefreq** = **nstep**/20
  - **backupfreq** = **nstep**/5
  - **runoutput** = 'full'
  - **pdb_output_freq** = **nstep**/5
  - **loutdft** = .FALSE.
  - **loutlammps** = .FALSE.
  - **loutdlpoly** = .FALSE.
  - **louthist** = .FALSE.
  - **pressurefreq** = 10, unless **ensemble** = 'uVT' in which case **pressurefreq** = 0
  - **trmaxdispfreq** = **nstep**/10
  - **volmaxdispfreq** = **nstep**/10
- 'manual': This is the default setting if no other value of **controlstyle** is specified. When using this option you must also include the following variables

  printfreq (integer) *optional parameter*
  - The step frequency for outputting information about the system to standard output (fortran unit 6). The information is the number of Monte Carlo steps performed thus far during the run, the total energy in each box, the x-box length of each box, the pressure of each box, and the number of molecules of each type in each box. If printfreq = 0, or if it is not specified, there

will be no output.

Variables in this subsection only need to be specified if **tmmc_flag** is .TRUE..

n_tmmc_min (integer) *optional*

- Minimum number of molecules allowed in the simulation box. By default, this is set to 0. Specifying a non-zero value is useful when sampling specific molecule number ranges. The maximum number is set to **nmolty**.

weight_freq (integer)

- The step frequency for updating the biasing function for transition-matrix Monte Carlo.

c_matrix_freq (integer)

- The step frequency for writing to file the collection matrix for TMMC. A new file indicating the step number will be created.

run_name (character)

- File prefix for TMMC output. The current estimate of the natural logarithm of the particle number probability distribution is written to a file "run_name.tmmc_weights.dat" every time the biasing function is updated. Collection matrix info is written to a file "run_name.c.stepnumber.dat". Accumulated semigrand potential energies are written to a file "run_name.vsg.stepnumber". This information is useful if one is interested in the average potential energy as a function density.

in_c_flag (logical) *optional parameter*

- By default, set to .FALSE. Set this variable to .TRUE. if an initial collection matrix is to be used to initiate the TMMC simulation. In the future, this will be modified so that either a collection matrix or biasing function can be read in. This option can be used to for continuing a TMMC simulation.

in_cfile (character) *optional parameter*

- Name of the collection matrix input file to be read in. Only needed if **in_c_flag** is .TRUE..

End of variables subsection when **tmmc_flag** is .TRUE..

---

blocksize (integer) *optional parameter*

- The size of the blocks for computing block averages. If you want this to be meaningful then blocksize should divide cleanly into nstep. The quantities that are averaged (in each simulation box) are the specific density, the pressure, all of the energy terms, the chemical potential of each molecule type, number density of each molecule type, and the mole fractions. If blocksize=0, or if it is not specified, no block output takes place.

moviefreq (integer) *optional parameter*

- The step frequency for outputting the system conformations to the towhee_movie file. This file is analyzed after the run using the analyze_movie.F routine to compute a variety of distribution functions. This file can get pretty big if you output frequently so be careful if you have a limited amount of hard disk space available. If moviefreq=0, or if it is not specified, this file is not written.

backupfreq (integer) *optional parameter*

- The step frequency for writing a file named towhee_backup that is suitable for use as a restart file. It overwrites the previous version of towhee_backup each time so it does not take up much space. Typically I set backupfreq so that I get around 10 backups during a run. For more information about restart files look at the manual entries for towhee_initial, towhee_backup, and towhee_final. If backupfreq=0, or if it is not specified, this file is not written.

restartfreq (integer) *optional parameter*

- The step frequency for writing a file named towhee_restart_NNN that is suitable for use as a restart file, where NNN represents the step number at which this file is written. This file is identical to towhee_backup (see **backupfreq** above), except that it is not overwritten by subsequent restart files, and allows for restarts from multiple points along a run. If restartfreq=0, or if it is not specified, this file is not written.

runoutput (character*20)

- 'full': if you want information about the individual blocks of the block averages and information about the maximum displacement updates.

- 'blocks': if you want information about the individual blocks of the block averages and don't want information about the maximum displacement updates.
- 'updates': if you don't want information about the individual blocks of the block averages and do want information about the maximum displacement updates.
- 'none': if you don't want information about the individual blocks of the block averages or information about the maximum displacement updates.

pdb_output_freq (integer) *optional parameter*

- The step frequency for outputting a snapshot of the simulation to a pdb file named box_xx_step_yyyyyyyyyyyyyy.pdb where xx is the box number converted into a 2 character string and yyyyyyyyyyyyyy is the step number converted into a 14 character string. If you do not wish to output any pdb files then you can set **pdb_output_freq** to 0 to disable this feature.

loutdft (logical)

- .TRUE. if you wish to output files for use with the Tramonto classical density functional theory code. This outputs dft_surfaces.dat and dft_decode.dat. See the Tramonto manual for information about these files.
- .FALSE. if you do not want to output dft files.

loutlammps (logical)

- .TRUE. if you wish to output files for use with the LAMMPS massively parallel molecular dynamics code. This outputs lammps_input and lammps_data# where the number is each of the simulation box numbers. See the LAMMPS manual for more information on how to read in these files.
- .FALSE. if you do not want to output LAMMPS files.

loutdlpoly (logical) *optional parameter*

- .TRUE. if you wish to output files for use with the DL_POLY molecular dynamics code. This outputs CONFIG# and FIELD# where the number is each of the simulation box numbers. See the DL_POLY manual for more information about these files.
- .FALSE. if you do not want to output DL_POLY files (default value).

The variables in this subsection are only included if **ensemble** is 'uvt'

louthist (logical)

- .TRUE. if you wish to output files used for histogram reweighting. When set to this value you must also include the following additional variables

    hist_label (integer)

    - An integer that is turned into a character string that creates the X portion of the named towhee_hisXY.dat file that is used to output information for histogram reweighting.

    hist_suffix (character*1)

    - A single character that creates the Y portion of the named towhee_hisXY.dat file that is used to output information for histogram reweighting.

    hist_nequil (integer)

    - The number of steps (cycles or moves) that are disregarded for the purposes of outputting histogram information. Set to 0 if you wish to use all of the steps for computing the histogram, and set to a positive number in order to discard those initial steps from the histogram.

    histcalcfreq (integer)

    - The step frequency for computing the information needed for histogram reweighting.

    histdumpfreq (integer)

    - The step frequency for outputting the information needed for histogram reweighting to the various towhee_histogram files. The ratio of **histdumpfreq/histcalcfreq** must be less than NDUMPHIST.

- .FALSE. if you do not wish to output files for histogram reweighting. No additional variables are required for this setting.

End of the subsection only included if **ensemble** is 'uvt'

---

pressurefreq (integer) *optional parameter*

- The step frequency for computing the pressure via the pressure virial (for continuous potentials) or the radial pressure (for discontinuous potentials) in each simulation box. Be aware that computing the pressure is a relatively expensive task (especially for large systems). If you do not wish to compute the pressure using these methods you can set the **pressurefreq** to zero to disable this calculation. If the parameter is not specified, the default value is 0.

trmaxdispfreq (integer) *optional parameter*

- The step frequency for updating the maximum translational (atom and center-of-mass) and rotational displacements. They are adjusted to try and achieve the target acceptance rates (see **tatraa**, **tatrac**, and **tarot**). It is a good idea to do this fairly frequently at the start of the simulation (every step or every 10 steps) in order to get good values for the maximum displacements. Once the acceptance rates are near their desired values then reset **trmaxdispfreq** to do 10 updates during a run. If **trmaxdispfreq** is 0, or if it is not specified, no updates take place.

volmaxdispfreq (integer) *optional parameter*

- The step frequency for updating the maximum volume displacements. They are adjusted to try and achieve the target acceptance rates (see **tavol**). It is a good idea to do this fairly frequently at the start of the simulation (every few steps) in order to get good values for the maximum displacements. Once the acceptance rates are near their desired values I typically set **volmaxdisp** to do 10 updates during a run. If **volmaxdispfreq** is set to 0, or if it is not specified, no such updates take place.

chempotperstep (integer)

- The number of additional trial insertions to perform in each box for at the end of every Monte Carlo step (listed sequentially for each molecule type on a single line). This allows the measurement of chemical potential in ensembles that do not have an insertion and deletion move (such as canonical and isobaric-isothermal).

potentialstyle

- 'internal': uses energy routines internal to the Towhee software package to describe the energies between atoms. When using this option you must include the following variables.

ffnumber (integer)

- 1 or more: reads the force field information from this number of file(s) listed in the **ff_filename**.

ff_filename (formatted character*70) [one line for each force field]

- A list of the filenames (one per line) that contain the force field information. On most systems you can just list this directory and then end the line. However, if you have trouble then adding sufficient blank spaces to the end of the line to get up to 70 characters could resolve your problem.

classical_potential (character*30)

The setting for this variable controls the nonbonded potential type. Depending on the setting there are a number of other variables that are also required. Please see the classical_potential web page for more information.

electrostatic_form (character*50)

- 'none' : no electrostatic potential
- 'coulomb' : use Coulomb's law for the electrostatic potential. This option requires the following variables.

coulombstyle (character*20)

- 'ewald_fixed_kmax' if you want to use point charges with an Ewald sum that utilizes a constant number of inverse space vectors (**kmax**) and a variable electrostatic cutoff (**rcelect**) equal to half the current box length. When using this option you also need the following variables.

kalp (double precision)

- Value used in the Ewald sum to compute alpha. The actual Ewald sum alpha term is equal to **kalp** divided by the shortest box length. The recommended value for **kalp** is 5.6.

kmax (integer)

- Maximum number of inverse space vectors to use in any dimension for the Ewald sum. Recommended value of this parameter is 5. If you want to set this to a larger value to may have to increase VECTORMAX.

dielect (double precision)

- The dielectric constant used when computing coulombic interactions. Generally this should be set to 1.0 as the solvated system will act as the screening that the dielectric constant is intended to represent. If you are performing a simulation without any solvent (for example a protein without the water) you might want to set this value to the dielectric constant of the missing solvent.

- 'ewald_fixed_cutoff' if you want to use point charges with an Ewald sum that utilizes a constant electrostatic cutoff (**rcelect**) and adjusts the number of inverse space vectors (**kmax**) according to the following heuristic.
alpha = (1.35 - 0.15 log[**ewald_prec**]) / **rcelect**
kmax = (alpha * Max[box length] / p) * (log[**ewald_prec**])$^{0.5}$
When using this option you will also need to list the following variables.

ewald_prec (double precision)

- Controls the precision of the Ewald summation technique. The smaller the value, the better the results (and the more expensive the simulation). The recommended value of 1d-4 is generally adequate, while a value of 1d-5 is very good (but more expensive).

rcelect (double precision)

- The cutoff for electrostatic interactions computed in the "real space" portion of the Ewald sum. Decreasing this value means less work in the "real space", but correspondingly more work in the "inverse space". Setting this equal to the general nonbonded cutoff (see **rcut** in classical_potential) is recommended.

dielect (double precision)

- The dielectric constant used when computing coulombic interactions. Generally this should be set to 1.0 as the solvated system acts as the screening that the dielectric constant is intended to represent. If you are performing a simulation without any solvent (for example a protein without the water) you could set this value to the dielectric constant of the missing solvent.

- 'minimum image' uses the minimum image convention to compute the coulombic interactions between all pairs of atoms in a system. Please note that this option is implemented mainly to compute single-molecule energies for test systems and is not suggested for routine use in periodic systems. For a discussion of the perils of using cut-off methods for Coulombic interactions please see Hummer *et al.* 1997. When using this option you must list the following variable.

dielect (double precision)

- The dielectric constant used when computing coulombic interactions. Generally this should be set to 1.0 as the solvated system acts as the screening that the dielectric constant is intended to represent. If you are performing a simulation without any solvent (for example a protein without the water) you could set this value to the dielectric constant of the missing solvent.

nfield (integer)

The number of external fields to apply in the simulation. These fields can take on a variety of forms, but are always applied relative to a plane in one of the simulation boxes. Typical uses are for simulating the effect of a rigid surface without having to treat the surface atoms explicitly. If nfield is set to anything other than 0 you will need to list the following variables for each field you wish to specify.

fieldtype (character*20)

- 'Hard Ball': Places a hard ball of a specified diameter at a particular location in one of the boxes. This ball interacts with each atom in the simulation and excludes all atom centers from its radius. When using this option you must also specify the following variables.
  - **hrbbox (integer)**
    - The number of the simulation box which contains this hard ball. Must range from 1 to **numboxes**.
  - **hrbpos (double precision)**
    - Position of the center of the hard ball. Must be between 0.0 and the box length of each of the 3 axes. All three coordinates are listed on the same line.
  - **hrbrad (double precision)**
    - Radius of the hard ball. The ball is felt through the periodic boundaries.
  - **hrbfac (double precision)**
    - Factor used to rescale the size of the hard ball. A value of 1.0 will use the **hrbrad** value without any adjustment. This factor is multiplied by the **hrbrad** value to compute the true radius of the hard ball.
  - **hrb_energy_type (character*11)**
    - 'infinite': for a hard overlap with the ball of infinite energy
    - 'finite': for a specfic finite penalty for each atom that is inside of the hard ball. When using this option you also need the following variable.
      - **hrb_ball_energy (double precision)**
        - The energy penalty in units of K/k_B for each atom that overlaps with a hard ball.
- 'Hard Wall': Places a hard wall of a specified diameter in one of the boxes. This wall interacts with each atom in the simulation. When using this option you must also specify the following variables.
  - **hrdbox (integer)**
    - The number of the simulation box which contains this hard wall. Must range from 1 to **numboxes**.
  - **hrdxyz (character*1)**
    - 'x': hard wall is perpendicular to the x-axis (in the yz plane)
    - 'y': hard wall is perpendicular to the y-axis (in the xz plane)
    - 'z': hard wall is perpendicular to the z-axis (in the xy plane)
  - **hrdcen (double precision)**
    - Position of the center of the hard wall. Must be between 0.0 and the box length of the axis that is perpendicular to the wall.
  - **hrdrad (double precision)**
    - Radius of the hard wall. The wall is felt through the periodic boundaries.
  - **hrd_repulsion_style (character*11)**
    - 'centers': the center of each atom in the system is the portion that interacts with the hard wall region. In other words, the hard wall interacts with the atoms as if they were point particles.
    - 'hard radii': the atoms are considered to have a radius that is determined by their nonbonded parameters. The distance between the surface of the hard wall and the closest approach of the atomic radius is used to determine the interactions between the wall and the atoms. This option is not currently supported for all choices of the **classical_potential**.
  - **hrd_energy_type (character*11)**
    - 'infinite': any molecule inside of the hard wall has an infinite energy (hard overlap).
    - 'finite': any molecule inside of the hard wall has a finite energy that is specified by the **hrd_wall_energy** variable. If you use this option you must include the following variable
      - **hrd_wall_energy (double precision)**
        - The energy given to any atom that is inside of the hard wall (in Kelvin). This option is designed to enable equilibrium of a hard

            wall system as this provides an incentive for molecules to leave the hard wall area without causing a simulation ending overlap.

- 'Harmonic Attractor': Uses a harmonic potential to root certain atoms to a defined point or their initial positions.

  hafbox (integer)
  - The number of the simulation box in which this harmonic attractor is employed. Must range from 1 to numboxes.

  hafk (double precision)
  - The force constant for the harmonic potential.

  hafentries (integer)
  - The number of types or elements to which this field is applied.

  hafrefpos (character*7)
  - 'Global': Uses a global set of coordinates for each atom. When using this option you also need to include the following variable.

    hafglobxyz (double precision array)
    - The x,y, and z coordinates of the global position. This should be entered all on the same line just separated by spaces.
  - 'Initial': Uses the initial coordinates of each atom.

  hafkey (character*7)
  - 'Element': Allows the user to chose to apply this field to a specific group of atoms which are all the same type of element. The following variables must be included for each entry.

    hafmolec (integer)
    - The field is applied to the element of choice in this molecule number.

    hafelement (character*2)
    - The element type to which apply this field.
  - 'FFtype': Allows the user to chose to apply this field to a specific group of atoms which are all the same nonbond type. The following variables must be included for each entry.

    hafmolec (integer)
    - The field is applied to the element of choice in this molecule number.

    hafname (character*10)
    - The nonbond type to which apply this field.
- 'Hooper Umbrella': Places a Hooper Umbrella field (see Hooper *et al.* 2000) in a simulation box. This is a 4th power energy function based on displacement along a single axis.
  v(d) = umba * [(d - umbcenter) / umbcenter]
  With this option you must also specify the following variables.

  umbbox (integer)
  - The number of the simulation box which contains this Umbrella field. Must range from 1 to numboxes.

  umbxyz (character*1)
  - 'x': Field is perpendicular to the x-axis (in the yz plane)
  - 'y': Field is perpendicular to the y-axis (in the xz plane)
  - 'z': Field is perpendicular to the z-axis (in the xy plane)

  umbcenter (double precision)
  - The zero energy point of the field, listed as a distance in Angstroms along the axis specified in **umbxyz**

  umba (double precision)
  - The energy constant in units of K/k_B.
- 'LJ 9-3 Wall': Places a 9-3 Lennard-Jones wall in one of the simulation boxes. The wall potential has the following functional form.

3 9 3

$v(d) = [\ 2/3\ p\ e_{wf}\ s_{wf}\ \ \rho_{wall}\ ] * [\ 2/15\ (s_{wf}/d)\ \ -\ (s_{wf}/d)\ \ ]$

With this option you must also specify the following variables.

- **ljfbox (integer)**
  - The number of the simulation box which contains this Lennard-Jones wall. Must range from 1 to numboxes.
- **ljfxyz (character*1)**
  - 'x': Lennard-Jones wall is perpendicular to the x-axis (in the yz plane)
  - 'y': Lennard-Jones wall is perpendicular to the y-axis (in the xz plane)
  - 'z': Lennard-Jones wall is perpendicular to the z-axis (in the xy plane)
- **ljfcen (double precision)**
  - Position of the center of the Lennard-Jones wall. Must be between 0.0 and the box length of the axis that is perpendicular to the wall.
- **ljfdir (integer)**
  - -1: Atoms only interact with the "left" face of this wall. This does not extend through the periodic boundary.
  - 1: Atoms only interact with the "right" face of this wall. This does not extend through the periodic boundary.
- **ljfcut (double precision)**
  - The distance beyond which the wall-atom interactions are not computed and assumed to be zero.
- **ljfshift (logical)**
  - T: if you want to shift the lj wall potential to be zero at the cutoff.
  - F: if you do not want to shift the potential.
- **ljfrho (double precision)**
  - The number density of atoms in the integrated wall potential (units of atoms per cubic Angstrom).
- **ljfntypes (integer)**
  - The number of atom types in the system that interact with the wall. Any atom type not listed here will have zero interaction with the wall. For each value of **ljfntypes** you must list the following variables.
    - **ljfname (character*6)**
      - The name of the atom. This must match up with the atom names listed in the **inpstyle**=2 portion of each molecule that is interacting with this wall. If you are not using that **inpstyle** this will still work except you will need to know the atom names in the appropriate towhee_ff_* files.
    - **ljfsig (double precision)**
      - s parameter for the interaction between this atom and the wall atoms. Units are Angstroms.
    - **ljfeps (double precision)**
      - e parameter for the interaction between this atom and the wall atoms. Units are K/k_B.

- 'Steele Wall': Places a 10-4 Lennard-Jones wall in one of the simulation boxes. The wall potential has the following functional form.

  $v(z) = e_w\ [\ 2/5\ (s_{sf}/z)^{10} - (s_{sf}/z)^4 - s_{sf}^4 / [\ 3\ \Delta\ (\ z + 0.61\ \Delta\ )^3\ ]\ ]$

  where

  $e_w = 2\ p\ e_{sf}\ \rho_s\ s_{sf}^2\ \Delta$

  This potential is attributed to Steele 1973, but I found that reference a bit confusing so I implemented the equations as presented in Lastoskie *et al.* 1993 and the variable names here follow the notation in that paper.

  With this option you must also specify the following variables.

  - **steele box (integer)**
    - The number of the simulation box which contains this Steele wall. Must range from 1 to numboxes.

steele xyz (character*1)
- 'x': wall is perpendicular to the x-axis (in the yz plane)
- 'y': wall is perpendicular to the y-axis (in the xz plane)
- 'z': wall is perpendicular to the z-axis (in the xy plane)

steele surface (double precision)
- Position of the surface of the wall. Must be between 0.0 and the box length of the axis that is perpendicular to the wall.

steele dir (integer)
- -1: Atoms only interact with the "left/bottom" face of this wall. This does not extend through the periodic boundary.
- 1: Atoms only interact with the "right/top" face of this wall. This does not extend through the periodic boundary.

steele cutoff (double precision)
- The distance beyond which the wall-atom interactions are not computed and assumed to be zero.

steele shift (logical)
- T: if you want to shift the wall potential to be zero at the cutoff.
- F: if you do not want to shift the potential.

steele delta (double precision)
- Δ parameter for the spacing between the layers in the solid. Units are in Angstroms.

steele rho_s (double precision)
- ρ_s parameter for the density of the atom in the solid. Units are in atoms per cubic Angstrom.

steele ntype (integer)
- The number of atom types in the system that interact with the wall. Any atom type not listed here will have zero interaction with the wall. For each type you must list the following variables.

  steele name (character*6)
  - The name of the atom. This must match up with the atom names listed in the **inpstyle**=2 portion of each molecule that is interacting with this wall. If you are not using that **inpstyle** this will still work except you will need to know the atom names in the appropriate towhee_ff_whatever files.

  sigma_sf (double precision)
  - s parameter for the interaction between this atom and the wall atoms. Units are Angstroms.

  epsilon_sf (double precision)
  - e parameter for the interaction between this atom and the wall atoms. Units are K/k_B.

- 'external': calls an external code (compiled as a library) that computes the total energy of the system. When using this option you need to include the following variable.

  external_code (character*20)
  - 'lcao': uses the lcao code of Peter Schultz, also known as Quest or SeqQuest, to compute the quantum mechanical energy. Use of this option requires the placement of several libraries into the /towheebase/lib directory (use of symbolic links is also acceptable in that directory). The code must be compiled with the '--enable-lcao' option in order to link in these libraries. When using this option you must also include the following variables.

    lcao_functional (character*20)
    - 'LDA': uses the local density approximation to compute the energy.

    lcao_atomtypes (integer)
    The number of different elements used in the quantum code.

    lcao_filename (character*50 array)
    A list of the atom filenames used in Quest. Listed one on a line for each atom type.

lcao_gridmultiplier (double precision)
Value that is multiplied by the box dimensions in order to compute the lcao grid dimensions. Suggested value is 4.0.

lcao_kgridproduct (double precision)
This value is divided by the box dimensions in order to compute the lcao kgrid. Suggested value is 40.0.

solvation_style (character*20)

- 'none' : no additioanl solvation potential for this system.
- 'internal' : use one of the solvation potentials internal to Towhee. This option requires the following variable.
  solvation_type (character*20)
  - 'EEF1' : solvation using the Charmm19-EEF1 potential.
- 'external' : use a solvation potential via an external program that has been compiled as a library and linked into Towhee.
  solvation_type (character*20)
  - 'tramonto' : solvation using the classical density functional theory code Tramonto to compute a solvation free energy.

linit (logical)

- .TRUE. if you are starting the simulation and wish to generate the positions of all of the atoms, assign initial box lengths and maximum displacements.
- .FALSE. if you want to continue the simulation by reading in box lengths, maximum displacements, and coordinates from towhee_initial.

initboxtype (character*20)

- 'unit cell': generates an initial structure by duplicating a unit cell. Reads information from the towhee_cell file and uses that to create an initial structure.
- 'dimensions': the dimensions of the initial boxes are entered in order to construct the initial boxes.
- 'number density': the total number density of molecules in the initial boxes are entered in order to compute the initial sizes of the boxes. This option generates cubic boxes.

initstyle (character*20)
This variable is only required for **initboxtype** settings of 'dimensions' or 'number density'
One line for each simulation box in the system. Each line contains a value for each molecule type.

- 'full cbmc': A template for this molecule type is created using configurational-bias. This template is then replicated throughout the simulation box to generate an initial configuration.
- 'template': A template for this molecule type is read from towhee_template. This template is then replicated throughout the simulation box to generate an initial configuration.
- 'coords': The coordinates for each atom are read from towhee_coords. This is useful if you are starting from a different file format (such as pdb), or have another code for building an initial configuration.
- 'nanotube': The coordinates for each atom are read from towhee_nanotube. This file is generated by the Towhee code if you use the inpstyle for carbon nanotubes. This template is then replicated throughout the simulation box to generate an initial conformation.
- 'helix cbmc': The molecule is generated by placing some of the backbone atoms onto a helix and then growing the rest of the atoms using CBMC. Any molecule initialized using this style must have the following information listed subsequent to the **initstyle** variables.
  helix_moltyp (integer)
  An integer corresponding to the molecule type that had an **initstyle** variable set to 4 in one of the simulation boxes. These must be listed in consecutive order.
  helix_radius (double precision)
  The radius of the helix (units of Angstroms).
  helix_angle (double precision)
  The pitch angle the helix makes with respect to the z-axis (units of degrees).
  helix_keytype (character*10)
  - 'element' compares the **helix_keyname** with the character*2 variable element that contains the 2 letter elemental code for each atom.
  - 'nbname' compares the **helix_keyname** with the character*10 variable nbname that contains the 10 character code for each atom type. This is the same variable that is used when inputting the

atom names with the Atom-based connectivity map (**inpstyle**=2).
  - 'pdbname' compares the **helix_keyname** with the character*4 variable pdbname that contains the 4 character code used in the pdb format output. This is most suitable for use with the Polypeptide builder (**inpstyle**=1) or the Nucleic acid builder (**inpstyle**=4).

helix_keyname (character*10)
The key for finding matches of the atom with the data structures for the molecule that is being grown as a helix. You need to choose an atom name that only appears in the backbone (e.g. 'P' for Charmm27 nucleic acids when using the element keytype, or ' CA ' for the C alpha backbone carbon of a polypeptide when using the pdbname keytype).

helix_conlen (double precision)
The distance between consecutive **helix_element** atoms (units of Angstroms).

helix_phase (double precision)
The initial angle of the helical chain (units of degrees). Normally, this has little effect as it is just a rotation about the z-axis, but if you are trying to set up two complementary nucleic acid chains to form a double helix then you would want their phase angles to differ by 180 degrees.

- 'partial cbmc': The molecule is constructed in two steps. First, a partial list of atom positions, amino acid 3-letter codes, and 4-letter atom identifiers (following the pdb standard) is read from towhee_partial and matched up against the expected amino acid and atom codes from the molecule template listed in towhee_input. Then configurational-bias is used to grow all of the missing atoms to create the initial structure.

initlattice (character*20)
This variable is only required for **initboxtype** settings of 'dimensions' or 'number density'.
One line for each simulation box in the system. Each line contains a value for each molecule type.
- 'center': puts the center of mass of the molecule in the very center of the box. This is a very poor idea if you have multiples of the same molecule type in a box as they will all be right on top of each other. However, it is useful if you are trying to get multiple different molecule types in an initial configuration where one molecule is inside of another (multi-walled nanotubes is a good example).
- 'none': place the molecules exactly where their template indicates. This option makes the most sense for **initstyle** options like 'coords' where you want to just read positions from a file.
- 'simple cubic': places the first atom of the molecule onto a simple cubic lattice and the rest of the atoms in the molecule are placed relative to that atom. If you are not using an **initstyle** of 'coords' then this is probably the option you want to use to generate an initial structure.

initmol (integer)
This variable is only required for **initboxtype** settings of 'dimensions' or 'number density'
- The initial number of each type of molecule in each box (one line per box).

inix, iniy, iniz (integer)
- The initial number of molecules (for **initboxtype** settings of 'dimensions' or 'number density') or of duplicated unit cells (for **initboxtype** setting of 'unit cell') in each direction in each box. The product of inix*iniy*iniz must be greater than or equal to the initial number of molecules in that box (for **initboxtype** settings of 'dimensions' or 'number density'). While these are labeled x, y, and z they actually correspond to the three coordinate vectors (truly x, y, and z for a rectangular box).

hmatrix (double precision)
This variable is only required for **initboxtype** settings of 'dimensions'
- The initial box dimensions (Angstroms) for the three vectors that describe the simulation box. There are nine entries (3 for each of the 3 vectors) in total for each simulation box. These are listed one vector at a time, with the three numbers which make up each vector listed on the same line. Note that the coordinate system you choose does not have to be orthogonal, but it must follow the right hand rule. The three vectors must also all be at least 45 degrees apart. Note that if you wish to use a rectangular box then only the diagonal elements of hmatrix will be non-zero, and these will be equal to the box lengths in the x, y, and z dimensions.

box_number_density (double precision)
This variable is only required for **initboxtype** settings of 'number density'
- The initial total number density of molecules in each simulation box. Listed as a single value per line with one line for each box as specified by **numboxes**. Units are molecules per nm^3.

- Note: the pm* variables are used to determine which move type to perform every time we want to do a Monte Carlo move. A move is selected by choosing a random number between 0.0 and 1.0 and then going down the list of pm* until you find one which has a value higher than the random number. At least one of the variables must be set to 1.0. A similar procedure is performed when we want to determine which boxes or molecule types to perform the selected move upon. These are done using the pm**pr and pm**mt arrays.

  Starting with Towhee release 4.15.2, all leading pm* probabilities are optional; if such a probability is not specified, then the rest of the variables in that section must likewise be left out, and the probability of that move is zero. As an example, if the variable **pmback** is not given, then **pmbkmt** must not be specified, and the *Configurational-Bias Protein Backbone Regrowth* move will never be performed.
- Comment: The formatting of the move variables is very specific. In all cases the first variable for a move (pm***) is left justified (as is the standard for most variable) while all other variables for that move are indented 10 spaces.

Isotropic Volume Move: These variables are only included for the following cases
- **ensemble** is 'npt'
- **ensemble** is 'nvt' and **numboxes** is 2 or more.

pmvol (double precision) *optional parameter*
- Probability of performing a volume move. If (**ensemble** is 'npt') then a single box is selected and it exchanges volume with an external pressure bath (see pressure). If (**ensemble** = 'nvt' and numboxes > 1) a pair of boxes are selected and volume is exchanged between them.

  pmvlpr (double precision)
  - Probability of performing a volume move on a particular box, or box pair. All of these variables are listed on a single line. If (**ensemble** = 'npt') then a value of pmvlpr is listed for each box. If (**ensemble** = 'nvt') then a value is listed for each pair of simulation boxes where the pairs are ordered (1,2), (1,3), ... (1,numboxes), (2,3), ... (numboxes-1,numboxes).

  rmvol (double precision) [a single value regardless of the actual number of box pairs]
  - The initial volume maximum displacement. If this is an isobaric-isothermal ensemble (**ensemble** = 'npt') then this is the initial maximum volume displacement (cubic Angstroms) in each box. If this is the canonical Gibbs ensemble (**ensemble** = 'nvt' and numboxes > 1) then this is the maximum displacement (logarithmic space) for each pair of boxes. As the simulation progresses, these values will be updated for each box, or each pair of boxes (see iratv).

  tavol (double precision)
  - The target acceptance rate for the volume move. Must be a value between 0.0 and 1.0. The volume displacement (rmvol) is periodically adjusted (see iratv) to yield this acceptance rate. I typically use a value of 0.5, though some researchers prefer smaller values.

Anisotropic Volume Move: These variables are only included for the following cases
- **ensemble** is 'npt'
- **ensemble** is 'nvt' and **numboxes** is 2 or more.

pmcell (double precision) *optional parameter*
- Probability of performing a unit cell adjustment move. If (**ensemble** = 'npt') then a single box is selected and a single hmatrix element is changed. This results in a volume exchange with a fictional external pressure bath (see pressure). If (**ensemble** = 'nvt' and numboxes > 1) a pair of boxes are selected. One of the boxes is then selected according to the pmcellpt variable and a single hmatrix element is changed in that box. This results in a change of volume for the first box which is countered by isotropically changing the volume in the second box.

  pmcellpr (double precision)
  - Probability of performing a unit cell adjustment move on a particular box, or box pair. All of these variables are listed on a single line If (**ensemble** = 'npt') then a value of pmvlpr is listed for each box. If (**ensemble** = 'nvt') then a value is listed for each pair of simulation boxes where the pairs are ordered (1,2), (1,3), ... (1,numboxes), (2,3), ... (numboxes-1,numboxes).

  pmcellpt (double precision)
  - Probability of selecting the first box of the pair as the box to perform the non-isotropic volume move upon, while its partner undergoes an isotropic volume move. This variable is only meaningful if (**ensemble** = 'nvt'). Note that you can choose to perform the non-isotropic volume move always on

the same box and this might be useful if you are doing a solid-vapor equilibria calculation.

rmcell (double precision)

- The initial unit cell adjustment maximum displacement. In all cases, this is the maximum amount (in Angstroms) that a single element of the hmatrix can change in a single unit cell move. Note, the in the canonical Gibbs ensemble case it is possible for the isotropic box to undergo an hmatrix change that is larger than this value as that box simply makes up for the volume change caused by the non-isotropic adjustment in the first box. As the simulation progresses, these values are updated for each box with a frequency controlled by **iratv**.

tacell (double precision)

- The target acceptance rate for the unit cell adjustment move. Must be a value between 0.0 and 1.0. The unit cell displacement (rmcell) is periodically adjusted (see iratv) to yield this acceptance rate. I typically use a value of 0.5.

Rotational-bias 2 box molecule Transfer Move: These variables are only included if **numboxes** is greater than or equal to 2

pm2boxrbswap (double precision) *optional parameter*

- Probability of performing a rotational-bias interbox molecule transfer move. This move takes a molecule out of one box and tries to place it in another box. The molecule is grown using **nch_nb_one** attempted different orientations and position (of the center-of-mass) for the new molecule.

  pm2rbswmt (double precision)

  - Probability of performing a rotational-bias interbox molecule transfer move on each type of molecule in the system.

  pm2rbswpr (double precision)

  - Probability of performing a rotational-bias interbox molecule transfer move between each pair of boxes in the system. The box pairs are ordered (1,2), (1,3), ... (1,numboxes), (2,3), ... (numboxes-1,numboxes).

Configurational-bias 2 box molecule Transfer Move: These variables are only included if **numboxes** is greater than or equal to 2

pm2boxcbswap (double precision) *optional parameter*

- Probability of performing a configurational-bias interbox molecule transfer move. This move takes a molecule out of one box and tries to place it in another box. The molecule is grown using coupled-decoupled configurational-bias Monte Carlo.

  pm2cbswmt (double precision)

  - Probability of performing a configurational-bias interbox molecule transfer move on each type of molecule in the system.

  pm2cbswpr (double precision)

  - Probability of performing a configurational-bias interbox molecule transfer move between each pair of boxes in the system. The box pairs are ordered (1,2), (1,3), ... (1,numboxes), (2,3), ... (numboxes-1,numboxes).

Configurational-bias grand-canonical insertion/deletion Move: These variables are only included if **ensemble** is 'uvt'

pmuvtcbswap (double precision) *optional parameter*

- Probability of performing a grand-canonical configurational-bias insertion or deletion move.

  pmuvtcbmt (double precision)

  - Probability of performing a grand-canonical configurational-bias insertion or deletion move on each type of molecule in the system.

Configurational-bias single box molecule Reinsertion Move

pm1boxcbswap (double precision) *optional parameter*

- Probability of performing an intrabox configurational-bias molecule transfer move. This move takes a molecule out of one box and tries to place it back into the same box. The molecule is grown using coupled-decoupled configurational-bias Monte Carlo.

  pm1cbswmt (double precision)

        - Probability of performing an intrabox configurational-bias molecule transfer move on each type of molecule in the system.

Intrabox two molecule switch based upon the center of mass positions
pm1boxcomswitch (double precision) *optional parameter*

- Probability of performing an intrabox exchange of the center of mass of two molecules of different types.
  pm1comswbox (double precision)
    - Probability of performing a center of mass switch move in each simulation box. List one value for each simulation box. At least one of the boxes must have a value of 1.0d0.

  pm1comswpair (double precision)
    - Probability of performing a center of mass switch move on each pair of molecule types in the simulation. List one value for each pair of molecule types in the simulation (**nmolty***(**nmolty**-1)/2). At least one of the pairs must have a value of 1.0d0.

Aggregation Volume Bias Move Type 1
pmavb1 (double precision) *optional parameter*

- Probability of performing an aggregation volume bias move of type 1, as described in Chen and Siepmann 2000. This is useful for forming and destroying clusters in simulations with molecules that tend to aggregate together.
  pmavb1in (double precision)
    - Probability of trying to move a molecule into an inner region for aggregation volume bias move of type 1.

  pmavb1mt (double precision)
    - Probability of performing an aggregation volume bias move of type 1 where a molecule of a certain type is moved. This is an array with one element for each molecule type.

  pmavb1ct (double precision)
    - Probability of performing an aggregation volume bias move of type 1 where the molecule target is of a certain type. The molecule that is moved is chosen according to **pmavb1mt** and then the type of molecule that is used as a reference for determining the inner and outer regions is found using this variable. This is a two dimensional array and uses one line of text for each type of molecule in the system.

  avb1rad (double precision)
    - The radius used to define the inner and outer volumes in the aggregation volume bias move of type 1. The distance is specified in Angstroms and must be greater than zero, but less than or equal to **rcut**.

Aggregation Volume Bias Move Type 2
pmavb2 (double precision) *optional parameter*

- Probability of performing an aggregation volume bias move of type 2, as described in Chen and Siepmann 2001. This is useful for forming and destroying clusters in simulations with molecules that tend to aggregate together.
  pmavb2in (double precision)
    - Probability of trying to move a molecule into an inner region for aggregation volume bias move of type 2.

  pmavb2mt (double precision)
    - Probability of performing an aggregation volume bias move of type 2 where a molecule of a certain type is moved. This is an array with one element for each molecule type.

  pmavb2ct (double precision)
    - Probability of performing an aggregation volume bias move of type 2 where the molecule target is of a certain type. The molecule that is moved is chosen according to **pmavb2mt** and then the type of molecule that is used as a reference for determining the inner and outer regions is found using this variable. This is a two dimensional array and uses one line of text for each type of molecule in the system.

  avb2rad (double precision)
    - The radius used to define the inner and outer volumes in the aggregation volume bias move of type 2. The distance is specified in Angstroms and must be greater than zero, but less than or equal to **rcut**.

Aggregation Volume Bias Move Type 3
pmavb3 (double precision) *optional parameter*

- Probability of performing an aggregation volume bias move of type 3, as described in Chen and Siepmann 2001. This is useful for transferring molecules between clusters.
  pmavb3mt (double precision)
  - Probability of performing an aggregation volume bias move of type 3 where a molecule of a certain type is moved. This is an array with one element for each molecule type.

  pmavb3ct (double precision)
  - Probability of performing an aggregation volume bias move of type 3 where the molecule target is of a certain type. The molecule that is moved is chosen according to **pmavb1mt** and then the type of molecule that is used as a reference for determining the inner and outer regions is found using this variable. This is a two dimensional array and uses one line of text for each type of molecule in the system.

  avb3rad (double precision)
  - The radius used to define the inner and outer volumes in the aggregation volume bias move of type 3. The distance is specified in Angstroms and must be greater than zero, but less than or equal to **rcut**.

Configurational-Bias Partial Molecule Regrowth
pmcb (double precision) *optional parameter*

- Probability of performing a molecule regrowth move on a molecule without regard to which box the molecule is currently located in. This move chooses a molecule of the appropriate type at random, selects an atom of the molecule at random, and then regrows the molecule either entirely (if a random number < pmall) or in all directions except for one. The molecule is regrown using configurational-bias.
  pmcbmt (double precision)
  - Probability of performing a molecule regrowth on each type of molecule in the system.

  pmall (double precision)
  - pmall is the probability that a molecule regrowth move will regrow the entire molecule. This is listed for each molecule type in the simulation.

Configurational-Bias Protein Backbone Regrowth
pmback (double precision) *optional parameter*

- Probability of performing configurational-bias fixed-endpoint regrowth of a portion of the protein backbone. This selects an atom along the peptide backbone, chooses another backbone atom that is connected by three bonds to the first atom, and then regrows all of the atoms in between these two atoms.
  pmbkmt (double precision)
  - Probability of performing a backbone regrowth move on each type of molecule in the system.

Configurational-Bias Peptide side-chain Regrowth
pmcbside (double precision) *optional parameter*

- Probability of performing a configurational-bias regrowth move on a single side-chain of peptide. This works by compiling a list of atoms that have a pdbname of ' CA ' and selecting one of those at the origin of a side-chain regrowth. This is functional for the normal amino acids, plus proline and disulfide bonded cysteines.
  pmcbsidemt (double precision)
  - Probability of performing a configurational-bias side-chain regrowth on each type of molecule in the system.

Torsional Pivot Move
pmpivot (double precision) *optional parameter*

- Probability of performing a pivot move about a random bond in the molecule. This move chooses a bond that is not entirely contained in a single ring structure, and has at least one bond emanating from each end, and then rotates one side of the molecule about that bond.
  pmpivmt (double precision)
  - Probability of performing a pivot move on each type of molecule in the system.

Concerted Rotation Move on a non-peptide backbone
pmconrot (double precision) *optional parameter*

- Probability of performing a concerted rotation move for a sequence of 9 atoms in a molecule.
  pmcrmt (double precision)
  - Probability of performing a concerted rotation move move on each type of molecule in the system.

Concerted Rotation Move over a 3 peptides backbone sequence
pmcrback (double precision) *optional parameter*

- Probability of performing a concerted rotation move on a sequence of three peptides in a polypeptide. This move only works for polypeptides.
  pmcrbmt (double precision)
  - Probability of performing a protein backbone concerted rotation move on each type of molecule in the system.

Plane Shift Move
pmplane (double precision) *optional parameter*

- Probability of performing a plane shift move. This move displaces all of the molecules whose center of mass lies in a plane of width **planewidth**. A new trial position for the center of the plane of atoms is generated uniformly across the available plane.
  pmplanebox (double precision)
  - Probability of performing a plane shift in each of the simulation boxes. List one value for each simulation box. At least one of the boxes must have a value of 1.0d0.

  planewidth (double precision)
  - The width of the plane for the plane shift move. Any molecule whose center of mass is within a plane of this thickness (whose position is chosen uniformly along one axis) will move during the plane shift move. The value of planewidth must be greater than 0.0d0 and less than the shortest box length.

Row Shift Move
pmrow (double precision) *optional parameter*

- Probability of performing a row shift move. This move displaces all of the molecules whose center of mass lies in a row of diameter **rowwidth**. A new trial position for the center of the row of atoms is generated uniformly across the available row.
  pmrowbox (double precision)
  - Probability of performing a row shift in each of the simulation boxes. List one value for each simulation box. At least one of the boxes must have a value of 1.0d0.

  rowwidth (double precision)
  - The width of the plan for the row shift move. Any molecule whose center of mass is within a row of this thickness (whose position is chosen uniformly along one axis) will move during the row shift move. The value of rowwidth must be greater than 0.0d0 and less than the shortest box length.

Intramolecular Single Atom Translation Move
pmtraat (double precision) *optional parameter*

- Probability of performing a single-atom translation move on a molecule without regard to which box the molecule is currently located in. This move chooses a molecule of the appropriate type at random, selects an atom of the molecule at random, selects a vector on a unit sphere at random, and then attempts to displace the atom a random distance between -rmtraa and +rmtraa in that direction.
  pmtamt (double precision)
  - Probability of performing a single-atom translation move on each type of molecule in the system.

  rmtraa (double precision)
  - The initial Atom-translation maximum displacement (Angstroms) for all molecules types in all boxes. As the simulation progresses, these values are updated to yield the desired acceptance rate for each molecule type in each box (see trmaxdispfreq).

  tatraa (double precision)
  - The target acceptance rate for the atom translation move. Must be a value between 0.0 and 1.0. The maximum atom translational displacement (rmtraa) is periodically adjusted (see trmaxdispfreq) to

yield this acceptance rate. I typically use a value of 0.5, though some researchers prefer smaller values.

Composite Move
pmcomposite (double precision) *optional parameter*

- Probability of performing a composite move that consists of a random center-of-mass translation and random rotation. This move is essentially a concatenation of the Center-of-Mass Molecule Translation Move and the Rotation about the Center-of-Mass Move.
  pmcomt (double precision)
  - Probability of performing a composite move on each type of molecule in the system.

  rmcomtra (double precision)
  - The molecular translation displacement (angstroms) for all molecule types in all boxes. This parameter is essentially identical to the **rmtrac** parameter for translations of molecular COM.

  rmcomrot (double precision)
  - The molecular rotation maximum displacement (radians) for all molecule types in all boxes. This parameter is essentially identical to the **rmrot** parameter for rotations about COM.

Center-of-Mass Molecule Translation Move
pmtracm (double precision) *optional parameter*

- Probability of performing a center-of-mass translation move on a molecule without regard to which box the molecule is currently located in. This move chooses a molecule of the appropriate type at random, chooses a vector on a unit sphere at random, and then attempts to displace the entire molecule a random distance between -rmtrac and +rmtrac in that direction.
  pmtcmt (double precision)
  - Probability of performing a center-of-mass translation move on each type of molecule in the system.

  rmtrac (double precision)
  - The initial Center-of-mass translation maximum displacement (Angstroms) for all molecule types in all boxes. As the simulation progresses, these values are updated to yield the desired acceptance rate for each molecule type in each box (see trmaxdispfreq).

  tatrac (double precision)
  - The target acceptance rate for the center-of-mass translation move. Must be a value between 0.0 and 1.0. The maximum center-of-mass translational displacement (rmtrac) is periodically adjusted (see trmaxdispfreq) to yield this acceptance rate. I typically use a value of 0.5, though some researchers prefer smaller values.

Rotation about the Center-of-Mass Move
pmrotate (double precision) *optional parameter*

- Probability of performing a rotation about the center-of-mass move for a molecule without regard to the box the molecule is currently located in. This move chooses a molecule of the appropriate type at random and then attempts to rotate the entire molecule about the x, y, and z axes that run through the center-of-mass a random number of radians between -**rmrot** and +**rmrot** around each of the three axes.
  pmromt (double precision)
  - Probability of performing a rotation move on each type of molecule in the system.

  rmrot (double precision)
  - The initial molecular rotation maximum displacement (radians) for all molecule types in all boxes. As the simulation progresses, these values are updated to yield the desired acceptance rate for each molecule type in each box (see trmaxdispfreq).

  tarot (double precision)
  - The target acceptance rate for the rotation move. Must be a value between 0.0 and 1.0. The rotation displacement (rmrot) is periodically adjusted (see trmaxdispfreq) to yield this acceptance rate. I typically use a value of 0.5, though some researchers prefer smaller values.

---

cbmc_style (character*30)

- 'coupled-decoupled': uses a combination of coupled and decoupled selections in order to perform the configurational-bias Monte Carlo moves. The general concepts of coupled and decoupled configurational-bias Monte Carlo are described in the main text of Martin and Siepmann 1999. Currently the only valid option in the code, but there are plans for additional options in the future. This setting also requires the following variable.

coupled_decoupled_form (character*30)

- 'Martin and Siepmann JPCB 1999': When performing a configurational-bias move use the coupled-decoupled formulation presented in the appendix of Martin and Siepmann 1999 with the addition of a decoupled bond selection.
- 'Coupled to pre-nonbond': Uses a new algorithm that is not yet published. The bond, bending, and dihedral selection are all decoupled from each other. However, they are all coupled to the pre-nonbond loop.

cbmc_setting_style (character*30)

- 'default ideal': sets up all of the configurational-bias variables according to a general set of default values using the 'ideal' generation styles. This option sets the following variables for all molecule types in the system. For more information about these variables see the explicit setting for this variable).

  cbmc_bond_generation = 'r^2 with bounds'
  cbmc_bend_generation = 'ideal'
  cbmc_dihedral_generation = 'ideal'
  cbmc_nb_one_generation = 'uniform'
  two_bond_fixed_endpoint_bias_style = 'analytic Boltzmann using angles and dihedrals'
  three_bond_fixed_endpoint_bias_style = 'analytic using max and min 2-4 distance'
  nch_nb_one = 10
  nch_nb = 10
  nch_pre_nb = 1
  nch_tor = 360
  nch_tor_connect = 360
  nch_bend_a = 1000
  nch_bend_b = 1000
  nch_vib = 1000
  vibrang = 0.85 1.15
- 'widom ideal': sets up all of the configurational-bias variables according to a general set of default values using the 'ideal' generation styles in such a manner that the chemical potential is computed correctly using the normal widom insertion. This option sets the following variables for all molecule types in the system. For more information about these variables see the explicit setting for this variable).

  cbmc_bond_generation = 'r^2 with bounds'
  cbmc_bend_generation = 'ideal'
  cbmc_dihedral_generation = 'ideal'
  cbmc_nb_one_generation = 'uniform'
  two_bond_fixed_endpoint_bias_style = 'none'
  three_bond_fixed_endpoint_bias_style = 'none'
  nch_nb_one = 1
  nch_nb = 1
  nch_pre_nb = 1
  nch_tor = 360
  nch_tor_connect = 360
  nch_bend_a = 1000
  nch_bend_b = 1000
  nch_vib = 1000
  vibrang = 0.85 1.15
- 'Martin and Thompson FPE 2004': sets up all of the configurational-bias variables to the values used in Martin and Thompson 2004. This option sets the following variables for all molecule types in the system. For more information about these variables see the explicit setting for this variable).

  cbmc_bond_generation = 'r^2 with bounds'
  cbmc_bend_generation = 'ideal'
  cbmc_dihedral_generation = 'ideal'
  cbmc_nb_one_generation = 'uniform'

two_bond_fixed_endpoint_bias_style = 'analytic Boltzmann using angles and dihedrals'
three_bond_fixed_endpoint_bias_style = 'analytic using max and min 2-4 distance'
nch_nb_one = 10
nch_nb = 10
nch_pre_nb = 1
nch_tor = 360
nch_tor_connect = 360
nch_bend_a = 100
nch_bend_b = 100
nch_vib = 1000
vibrang = 0.85 1.15

- 'default autofit gaussian': sets up all of the configurational-bias variables to autofit gaussian settings. This option sets the following variables for all molecule types in the system. For more information about these variables see the explicit setting for this variable).
  cbmc_bond_generation = 'autofit gaussian'
  bond_sdev_multiplier = 1.0d0
  cbmc_bend_generation = 'autofit gaussian'
  bend_a_sdev_multiplier = 1.0d0
  bend_b_sdev_multiplier = 1.0d0
  cbmc_dihedral_generation = 'autofit gaussian'
  dihedral_sdev_multiplier = 1.0d0
  cbmc_nb_one_generation = 'uniform'
  two_bond_fixed_endpoint_bias_style = 'autofit gaussian'
  two_bond_bias_sdev_multiplier = 1.0d0
  two_bond_bias_vibrange = 0.5d0 1.5d0
  three_bond_fixed_endpoint_bias_style = 'autofit gaussian using max and min 2-4 distance'
  three_bond_bias_sdev_multipler = 1.0d0
  nch_nb_one = 10
  nch_nb = 10
  if **coupled_decoupled_form** = 'Martin and Siepmann JPCB 1999' then
    nch_pre_nb = 1
    nch_tor = 10
    nch_tor_connect = 10
  else if **coupled_decoupled_form** = 'Coupled to pre-nonbond' then
    nch_pre_nb = 10
    nch_tor = 1
    nch_tor_connect = 1
  nch_bend_a = 1
  nch_bend_b = 1
  nch_vib = 1
- 'Martin and Frischknecht': sets up the configurational-bias values according to the strategies described in Martin and Frischknecht 2006. This option sets the following variables for all molecule types in the system. For more information about these variables see the explicit setting for this variable.
  cbmc_bond_generation = 'autofit gaussian'
  bond_sdev_multiplier = 1.0d0
  cbmc_bend_generation = 'ideal + autofit gaussian'
  bend_a_sdev_multiplier = 1.0d0
  bend_b_sdev_multiplier = 1.0d0
  bend_a_ideal_fraction = 0.01d0
  bend_b_ideal_fraction = 0.01d0
  cbmc_dihedral_generation = 'ideal + autofit gaussian'
  dihedral_peak_weight_style = 'uniform'
  dihedral_sdev_multiplier = 1.0d0
  dihedral_ideal_fraction = 0.01d0
  cbmc_nb_one_generation = 'uniform'

two_bond_fixed_endpoint_bias_style = 'analytic Boltzmann using angles'
three_bond_fixed_endpoint_bias_style = 'analytic using max and min 2-4 distance'
nch_nb_one = 10
nch_nb = 10
if **coupled_decoupled_form** = 'Martin and Siepmann JPCB 1999' then
  nch_pre_nb = 1
  nch_tor = 100
  nch_tor_connect = 100
  nch_bend_b = 100
else if **coupled_decoupled_form** = 'Coupled to pre-nonbond' then
  nch_pre_nb = 100
  nch_tor = 1
  nch_tor_connect = 1
  nch_bend_b = 1
nch_bend_a = 1
nch_vib = 1

- 'explicit': All of the configurational-bias options are explicitly required. When using this option you also need to include the following variables.

  cbmc_nb_one_generation (character*30 array) [one line for each simulation box and each line contains one value for each molecule type]

  - 'uniform' : the trial coordinates of first atom inserted into this simulation box are generated uniformly
  - 'energy bias' : the trial coordinates of the first atom inserted into this simulation box are generated by selecting a subvolume according to the Boltzmann weighted energy of a test atom in the center of each subvolume, averaged over all possible insertion atoms in the mapped molecule. Once a subvolume is selected then a position in that subvolume is generated uniformly. This biasing scheme was first used by Snurr *et al.* 1993. Note that the energy map used in practice is a linear combination of the Boltzmann weighted map described above (0.99 of the fraction) with a random selection of any subcube in the box (0.01 of the fraction) to make sure that the generation probability is strictly positive in all subcubes. The insertion energies are computed using the initial positions of all molecules in simulation box 1. If you are using this to help with insertions in a porous solid then you may want to only compute this for the empty porous solid and use the restart of that map in future simulations. When using this option you need the following variables.

    mapmolty (integer)

    - The molecule type to use as the insertion probe for determining the energy biasing weights.

    lcreatemap (logical)

    - .true. if you wish to create a new towhee_map file. This file contains information about the energy profile of the porous molecule. When using this setting you must also include the following additional variables

      cubex,cubey,cubez (integer)

      The number of cubelets you want to have in each direction of the box. Each dimension of the box is divided by the corresponding cube* value and for each cube an energy value is computed. This value is then used for biasing insertion/deletion moves. cubex*cubey*cubez mustbe less than or equal to MAXCUBE
    - .false. if you want to use a previously generated map in the towhee_map file. That file must have been generated using version 5.2.4 or later as there was a different, and problematic, format prior to that version.

  nch_nb_one (integer) [one value for each molecule type]

  - The number of trial positions that are sampled for the first atom inserted during a configurational-bias or rotational-bias molecule exchange move (see pm2boxrbswap, pm2boxcbswap, and pm1boxcbswap). The value must be less than or equal to NCHNB_MAX

  nch_nb (integer) [one value for each molecule type]

- The number of trial positions that are sampled for all atoms except for the first atom inserted during a configurational-bias molecule exchange move (see pm2boxcbswap and pm1boxcbswap). This is used for all atoms in a configurational-bias regrowth move. The value must be less than or equal to NCHNB_MAX.

nch_pre_nb (integer) [one value for each molecule type]

This variable is only required when the **coupled_decoupled_form** is 'Coupled to pre-nonbond'

- The number of trials for a selection procedure that takes place after the dihedral selection, but before the nonbond selection when using a **coupled_decoupled_form** of 'Coupled to pre-nonbond'. The value must be less than or equal to NCHTOR_MAX.

cbmc_dihedral_generation (character*30)

- 'ideal': dihedral trials are generated according to the ideal distribution. For dihedrals that means trials are generation uniformly on (-Pi, Pi).
- 'global gaussian': dihedral trials are generated according a series of gaussian distributions that are a function of the **bondpatt** on the central two atoms. This series is hard coded into Towhee and details can be found by looking in the getcbdihed.F subroutine. This gaussian bias is then removed in the acceptance rule. When using this option you must also include the following variable.

  sdevtor (double precision)

  The standard deviation (with units of degrees) that is used for each of the gaussian distributions for the dihedral angles. For best results this number should be set to the observed distribution computed in the analyse_movie utility. This number must be positive and a default value of 20.0 is suggested.
- 'autofit gaussian': dihedral trials are generated according to a series of gaussian distributions that are individually fit to the Boltzmann factor as a function of dihedral angle for each individual dihedral in the system. This fit is performed automatically at the start of each simulation using the equilibrium bond lengths and bending angles. This bias is then removed in the acceptance rule. When using this option you must also include the following variable.

  dihedral_sdev_multiplier (double precision)

  The factor that is multiplied by the observed gaussian standard deviation for each peak in the automatic dihedral fit in order to create the standard deviations that are used to generate the dihedrals during the simulation. This number must be positive and a default value of 1.0 is suggested.
- 'ideal + autofit gaussian': dihedral trials are generated according to a linear combination of the ideal distribution (uniform) and a series of gaussian distributions that are individually fit to the Boltzmann factor as a function of dihedral angle for each individual dihedral in the system. This fit is performed automatically at the start of each simulation using the equilibrium bond lengths and bending angles. This bias is then removed in the acceptance rule. When using this option you must also include the following variables.

  dihedral_peak_weight_style (character*30)

  - 'uniform': each peak is selected with the same (uniform) probability.
  - 'isolated Boltzmann': the peaks are selected with a probability proportional to the sum of the Boltzmann weight computed during the automatic fitting process.

  dihedral_sdev_multiplier (double precision)

  The factor that is multiplied by the observed gaussian standard deviation for each peak in the automatic dihedral fit in order to create the standard deviations that are used to generate the dihedrals during the simulation. This number must be positive and a default value of 1.0 is suggested.

  dihedral_ideal_fraction (double precision)

  The fraction of dihedral trials that are generated using the ideal distribution. The remainder of the trials are generated using the autofit gaussians. This number must be in the range [0.0,1.0] inclusive.

nch_tor (integer) [one value for each molecule type]

- The number of trial dihedral angles that are sampled during configurational-bias moves. The value must be positive and also less than or equal to NCHTOR_MAX.

nch_tor_connect (integer) [one value for each molecule type]

- The number of trial dihedral angles that are sampled during configurational-bias moves when we have grown the molecule such that we need to connect back up with atoms that already exist. This is needed in order to regrow cyclic molecules, and to regrow the interiors of large molecules. The value must be positive and also less than or equal to NCHTOR_MAX.

cbmc_bend_generation (character*30)

- 'ideal': bending trials are generated according to the ideal distribution. This is the Sine distribution for bending angle type A and uniform on (-Pi, Pi) for bending angle type B.
- 'global gaussian': bending trials are generated according to gaussian distributions. Bending A trials are generated according to a single gaussian with a mean set to the equilibrium bending angle and a standard deviation set to **sdevbena**. Bending B trials are generated according to one or more gaussian distributions with means set based upon the **bondpatt** of the central atom and a standard deviation set to **setbenb**. For more details on the bending B distribution see the getcbangle.F subroutine. When using this option you must also include the following variables.

  sdevbena (double precision)

  The standard deviation to use when generating the part A bending trials (units of degrees). Must be positive, and it is best to set to the observed distribution of the angles as measured by the analyse_movie utility.

  sdevbenb (double precision)

  The standard deviation to use when generating the part B bending trials (units of degrees). Must be positive, and it is best to set to the observed distribution of the angles as measured by the analyse_movie utility.
- 'autofit gaussian': bending trials are generated according to gaussian distributions. Bending A trials are generated according to a single gaussian with a mean and standard deviation fit to Sin(theta)*exp(-beta U_{bend}). The standard deviation used to generate trials is a product of the observed distribution standard deviation times **bend_a_sdev_multiplier**. Bending B trials are generated according to one or more gaussian distributions with means fitted to exp(-beta u_{bend}) for rotating the angles about a cone (with everything else set to the equilibrium bond lengths and bending angles). The observed standard deviation from this fit is multiplied by **bend_b_sdev_multiplier** to get the standard deviation used to generate bending B angles during the simulation. This bias is removed in the acceptance rules. When using this option you must also include the following variables.

  bend_a_sdev_multiplier (double precision)

  This value is multiplied by the observed standard deviation from performing the fit in order to create the standard deviation that is used to generate bending A trials during the simulation. This value must be positive and the currently suggested value is 1.0.

  bend_b_sdev_multiplier (double precision)

  This value is multiplied by the observed standard deviation from performing the fit in order to create the standard deviation that is used to generate bending B trials during the simulation. This value must be positive and the currently suggested value is 1.0.
- 'ideal + autofit gaussian': bending trials are generated according to a linear combination of the ideal distributions (sine for bend A, uniform for bend B) and the autofit gaussian distributions. Gaussian bending A trials are generated according to a single gaussian with a mean and standard deviation fit to Sin(theta)*exp(-beta U_{bend}). The standard deviation used to generate trials is a product of the observed distribution standard deviation times **bend_a_sdev_multiplier**. Gaussian bending B trials are generated according to one or more gaussian distributions with means fitted to exp(-beta u_{bend}) for rotating the angles about a cone (with everything else set to the equilibrium bond lengths and bending angles). The observed standard deviation from this fit is multiplied by **bend_b_sdev_multiplier** to get the standard deviation used to generate bending B angles during the simulation. This bias is removed in the acceptance rules. When using this option you must also include the following variables.

  bend_a_sdev_multiplier (double precision)

  This value is multiplied by the observed standard deviation from performing the fit in order to create the standard deviation that is used to generate bending A trials during the simulation. This value must be positive and the currently suggested value is 1.0.

bend_b_sdev_multiplier (double precision)

This value is multiplied by the observed standard deviation from performing the fit in order to create the standard deviation that is used to generate bending B trials during the simulation. This value must be positive and the currently suggested value is 1.0.

bend_a_ideal_fraction (double precision)

The fraction of bending A trials that are generated using the ideal distribution of Sin(theta). The remainder of the trials are generated using the autofit gaussians. This value must be in the range [0.0,1.0] inclusive.

bend_b_ideal_fraction (double precision)

The fraction of bending B trials that are generated using the ideal distribution of uniform on (-Pi,Pi). The remainder of the trials are generated using the autofit gaussians. This value must be in the range [0.0,1.0] inclusive.

nch_bend_a (integer) [one value for each molecule type]

- The number of trial angles that are sampled during configurational-bias moves when we are selecting the iugrow-iufrom-iuprev angle. This value must be positive. Currently suggested values are in the range of 100 to 1000 when using **cbmc_bend_generation** style 'ideal' and in the range from 1 to 10 when using **cbmc_bend_generation** styles 'global gaussian' or 'autofit gaussian'.

nch_bend_b (integer) [one value for each molecule type]

- The number of trial angles that are sampled during configurational-bias moves when we are selecting the rotation about a cone of one of the iugrow angles relative to the others. This value must be positive. Currently suggested values are in the range of 100 to 1000 when using **cbmc_bend_generation** style 'ideal' and in the range from 1 to 10 when using **cbmc_bend_generation** styles 'global gaussian' or 'autofit gaussian'.

cbmc_bond_generation (character*30)

- 'r^2 with bounds': Generate trial bond lengths according to a bounded r^2 probability distribution within the ranges set by the **vibrang** variable. This distribution is proportional to the true distribution, but has a limited sampling range while the true distribution is of infinite extent When using this option you also need to include the following variable.

  vibrang (double precision, double precision)

  The range of bond lengths to sample via configurational-bias Monte Carlo. The range is expressed as a fraction of the equilibrium bond length for the lower bound and the upper bound. Currently suggested values are 0.85 and 1.15.

- 'global gaussian': Generate trial bond lengths according to a gaussian distribution with a mean set to the equilibrium bond length and a standard deviation specified as **sdevvib**. When using this option you must also include the following variable.

  sdevvib (double precision)

  The standard deviation of a gaussian distribution that is used to sample bond lengths during a configurational-bias regrowth for a **cbmc_bond_generation** style of 'global gaussian'. Units are Angstroms. For best results perform a short simulation of single-atom translation moves, analyse that data using the analyse_movie utility, and set this value to the observed standard deviations in the bond length distribution.

- 'autofit gaussian': Generate trial bond lengths according to a gaussian distribution with a mean and standard deviation fitted to $r^2 \exp^{(-\text{beta } U_{\text{bond}})}$. When using this option you also need to include the following variable.

  bond_sdev_multiplier (double precision)

  This value is multiplied by the observed standard deviation of the $r^2 \exp^{(-\text{beta } U_{\text{bond}})}$ distribution in order to determine the standard deviation used to generate bond trials. This value must be positive. The currently suggested value is 1.0.

nch_vib (integer) [one value for each molecule type]

- The number of trial bond lengths that are sampled during a configurational-bias move. This value must be positive. Currently suggested values are 1000 trials for **cbmc_bond_generation** style 'r^2 with bounds' and a value in the range of 1 to 10 for **cbmc_bond_generation** styles 'global gaussian' and 'autofit gaussian'.

two_bond_fixed_endpoint_bias_style (character*50)

- 'none': No additional biasing is utilized during a configurational-bias move step that involves the growth of a new atom that is separated from an already existing atom by two bonds where the atom between those two bonds does not currently exist.
- 'analytic Boltzmann using angles': Additional biasing is utilized during a configurational-bias move step that involves the growth of a new atom that is separated from an already existing atom by two bonds where the atom between those two bonds does not currently exist. The biasing has the following form where i is the atom being grown, k is an atom that already exists, and j is an atom that is bonded to i and j, but has not yet been grown.
  $p_{bias}(r_{ik})$ = Min[minbias,p_{aBua}].
  p_{aBua}
  = p_{angle} for $r_{ik} < r_{ij}^0 + r_{jk}^0$
  = p_{bond} for $r_{ij}^0 + r_{jk}^0 < r_{ik}$
  p_{angle} = Exp[- ß u_{angle}(θ_{ijk})]
  where θ_{ijk} is computed from the trial r_{ik} distance and the equilibrium bond lengths of the two missing bonds (r_{ij}^0 and r_{jk}^0).
  p_{bond} = Exp[- ß (u_{angle}(θ_{ijk}) + u_{bond}(r_{ij}) + u_{bond}(r_{jk}))]
  where θ_{ijk} = p, $r_{ij} = r_{ij}^0 * r_{ik} / (r_{ij}^0 + r_{jk}^0)$, and $r_{jk} = r_{jk}^0 * r_{ik} / (r_{ij}^0 + r_{jk}^0)$,
  minbias is a minimum value set in the code in order to avoid division by zero. This is set in the febias.F subroutine and currently has a value of 1.0d-40.
- 'analytic Boltzmann dihedral energy sum': Additional biasing is utilized during a configurational-bias move step that involves the growth of a new atom that is separated from an already existing atom by two bonds where the atom between those two bonds does not currently exist. The biasing has the following form where i is the atom being grown, h is an atom that already exists and is the atom from which i is being grown this step, k is an atom that already exists, and j is an atom that is bonded to i and j, but has not yet been grown.
  $p_{bias}(r_{ij})$ = Min[minbias,p_{aBdes}].
  p_{aBdes}
  = $p_{dihedral}$ for $r_{ik} < r_{ij}^0 + r_{jk}^0$
  = minbias for $r_{ij}^0 + r_{jk}^0 < r_{ik}$
  $p_{dihedral}$ = Exp[- ß ($u_{dihedral}(f_{hijk}(1)) + u_{dihedral}(f_{hijk}(2))$)]
  where $f_{hijk}(1)$ and $f_{hijk}(2)$ are the two possible solutions for that dihedral given the following constraints,
  $r_{ij} = r_{ij}^0$
  $r_{jk} = r_{jk}^0$
  $\theta_{hij} = \theta_{hij}^0$
- 'autofit gaussian': Additional biasing is utilized during a configurational-bias move step that involves the growth of a new atom that is separated from an already existing atom by two bonds, where the atom between those two bonds does not currently exist. The biasing is a gaussian distribution that is fit to the following distribution where i is the atom being grown, k is an atom that already exists, and j is an atom that is bonded to i and j, but has not yet been grown.
  $p_{ag} = r_{ij}^2\ r_{jk}^2\ \theta_{ijk}$ Exp[-ß ($u_{bond}(r_{ij}) + u_{bond}(r_{jk}) + u_{angle}(\theta_{ijk})$)]

where this distribution is computed at the beginning of the simulation and then the mean and standard deviations are stored for use during the configurational-bias growth procedure. When using this option you must also include the following additional variables.

two_bond_bias_sdev_multiplier (double precision)

- This value is multiplied by the standard deviation determined from the fit in order to get the standard deviation that is used during the simulation. Must be positive.

two_bond_bias_vibrange (double precision array)

- These are the lower and upper bounds of sampling for the r_{ij} and r_{jk} bond lengths that are varied at the start of the simulation in order to determine the gaussian fit parameters. They are in units relative to the equilibrium bond lengths. Suggested default values are 0.5 and 1.5.

- 'self adapting gaussian using 1-3 distance': Additional biasing is utilized during a configurational-bias move step that involves the growth of a new atom that is separated from an already existing atom by two bonds, where the atom between those two bonds does not currently exist. The biasing is a gaussian distribution based upon the distance between the atom being grown and the target atom that is two bonds away and already exists. This gaussian distribution is self adapted during the course of a simulation so that it represents the observed 1-3 distance distribution. When using this option you must also include the following additional variables.

two_bond_bias_sdev_multiplier (double precision)

- This value is multiplied by the standard deviation determined from the fit in order to get the standard deviation that is used during the simulation. Must be positive.

two_bond_bias_initial_value (character*50)

- 'file': the initial distribution is read from the 'towhee_safe_initial' file in the local directory. This file is generally copied from the 'towhee_safe_final' file that is produced at the end of a simulation that employed this **two_bond_fixed_endpoint_bias_style**.
- 'autofit gaussian': the initial distribution is generated by automatically fitting a gaussian to a sampling of the r_{ik} distance. This option is recommended when starting a new simulation where there is no appropriate 'towhee_safe_initial' file available.

two_bond_bias_compute_frequency (integer)

- Statistics on the two bond bias distributions are taken from the simulation with a step frequency equal to this value. These statistics are then periodically used to update the distributions, as described in the **two_bond_bias_update_frequency** section. Set this value to 0 if you wish to disable the periodic computation of these distributions.

two_bond_bias_update_frequency (integer)

- The distributions used to perform the two bond biasing are updated with this step frequency. The updates combine the distributions computed with a frequency controlled by the **two_bond_bias_compute_frequency** variable with the previous version of the two bond biasing potentials to create the new two bond biasing potential. Set to 0 to disable this update.

two_bond_bias_old_fraction (double precision)

- This factor determines the linear combination of the old distribution, and the observed distribution, to determine the new two bond bias distribution. This number must be in the range [0.0,1.0] inclusive. Setting this value to 0.0 would completely replace the old distribution with the new distribution, while a setting of 0.5 would combine the old and observed distributions equally in order to compute the new distribution.

three_bond_fixed_endpoint_bias_style (character*50)

- 'none': No additional biasing is utilized during a configurational-bias move step that involves the growth of a new atom that is separated from an already existing atom by three bonds where the atoms between those three bonds do not currently exist.

'analytic using max and min 2-4 distance': Additional biasing is utilized during a configurational-bias move step that involves the growth of a new atom that is separated from an already existing atom by three bonds where the atoms between those three bonds do not currently exist. The biasing has the following form where i is the atom being grown, h is an atom that already exists and is the atom from which i is being grown this step, l is an atom that already exists, and j and k are atoms that have not yet been regrown and bridge the gap between atoms i and l. The biasing with this option depends upon the minimum (r_{jl}^{min}) and maximum (r_{jl}^{max}) projected distances between the j and l atoms given the constraints that r_{ij} and θ_{hij} are set to their equilibrium values (r_{ij}^0 and θ_{hij}^0). These minimum and maximum distances are then compared with the equilibrium distance between the j and l atoms (r_{jl}^{eq}). This equilibrium distance is computed by setting $r_{jk} = r_{jk}^0$, $r_{kl} = r_{kl}^0$, and $\theta_{jkl} = \theta_{jkl}^0$). The biasing value is computed as follows.

$p_{aumam24d}$

$= p_{stretch}$ for $r_{jl}^{eq} < r_{jl}^{min}$

$= 1.0$ for $r_{jl}^{min} < r_{jl}^{eq} < r_{jl}^{max}$

$= p_{compress}$ for $r_{jl}^{max} < r_{jl}^{eq}$

$p_{stretch}$

$=$ Exp[- ß $u_{angle}(\theta_{jkl})$] where θ_{jkl} is computed given the constraints of r_{jk}^0, r_{kl}^0, and r_{jl}^{min} for r_{jl}^{min} & lt; $r_{jk}^0 + r_{kl}^0$.

$=$ Exp[- ß ($u_{angle}(\theta_{jkl}) + u_{bond}(r_{jk}) + u_{bond}(r_{kl})$)] where $\theta_{jkl} = p$, $r_{jk} = r_{jk}^0 * r_{jl}^{min} / (r_{jk}^0 + r_{kl}^0)$, and $r_{kl} = r_{kl}^0 * r_{jl}^{min} / (r_{jk}^0 + r_{kl}^0)$ for $r_{jk}^0 + r_{kl}^0 < r_{jl}^{min}$.

$p_{compress}$ = Exp[- ß $u_{angle}(\theta_{jkl})$]

where θ_{jkl} is computed using the constraints r_{jk}^0, r_{kl}^0, and r_{jl}^{max}.

- 'autofit gaussian using max and min 2-4 distance': Additional biasing is utilized during a configurational-bias move step that involves the growth of a new atom that is separated from an already existing atom by three bonds where the atoms between those three bonds do not currently exist. The biasing has the following form where i is the atom being grown, h is an atom that already exists and is the atom from which i is being grown this step, l is an atom that already exists, and j and k are atoms that have not yet been regrown and bridge the gap between atoms i and l. The biasing with this option depends upon the minimum (r_{jl}^{min}) and maximum (r_{jl}^{max}) projected distances between the j and l atoms given the constraints that r_{ij} and θ_{hij} are set to their equilibrium values (r_{ij}^0 and θ_{hij}^0). These extrema are then combined with the 'autofit gaussian' biasing described for the **two_bond_fixed_endpoint_bias_style** option. The bias probability is set to the integrated probability of that autofit gaussian on the limits between r_{jl}^{min} and r_{jl}^{max}. When using this option you also need to include the following variable.

  three_bond_bias_sdev_multiplier
  - This value is multiplied by the standard deviation determined from the fit in order to get the standard deviation that is used during the simulation. Must be positive.
- 'self adapting gaussian using 1-4 distance': Additional biasing is utilized during a configurational-bias move step that involves the growth of a new atom that is separated from an already existing atom by three bonds where the atoms between those three bonds do not currently exist. The biasing has the following form where i is the atom being grown, h is an atom that already exists and is the atom from which i is being grown this step, l is an atom that already exists, and j and k are atoms that have not yet been regrown and bridge the gap between atoms i and l. The biasing is a gaussian that depends only upon the r_{il} distance. This

gaussian distribution can be set to self-adapt during the course of the simulation so that the biasing probability reflects the observed probability for the r_{il} distances. When using this option you also need to include the following variables.

three_bond_bias_sdev_multiplier

- This value is multiplied by the standard deviation determined from the fit in order to get the standard deviation that is used during the simulation. Must be positive.

three_bond_bias_initial_value (character*50)

- 'file': the initial distribution is read from the 'towhee_safe_initial' file in the local directory. This file is copied from the 'towhee_safe_final' file that is generated at the end of a simulation run that employed the appropriate **three_bond_fixed_endpoint_bias_style**.
- 'autofit gaussian': the initial distribution is automatically fit to the Boltzmann weight as a function of r_{il} distances using the equilibrium r_{ij}, r_{jk}, and r_{kl} bond lengths and the equilibrium θ_{ijk} and θ_{jkl} angles. This option is recommended for the initial simulation when no appropriate 'towhee_safe_initial' file exists.

three_bond_bias_compute_frequency (integer)

- Statistics on the three bond bias distributions are taken from the simulation with a step frequency equal to this value. These statistics are then periodically used to update the distributions, as described in the **three_bond_bias_update_frequency** section. Set this value to 0 if you wish to disable the periodic computation of these distributions.

three_bond_bias_update_frequency (integer)

- The distributions used to perform the three bond biasing are updated with this step frequency. The updates combined the distributions computed with a frequency controlled by the **three_bond_bias_compute_frequency** variable with the previous version of the three bond biasing potentials to create the new three bond biasing potential. Set to 0 to disable this update.

three_bond_bias_old_fraction (double precision)

- This factor determines the linear combination of the old distribution, and the observed distribution, to determine the new three bond bias distribution. This number must be in the range [0.0,1.0] inclusive. Setting this value to 0.0 would completely replace the old distribution with the new distribution, while a setting of 0.5 would combine the old and observed distributions equally in order to compute the new distribution.

The final section of towhee_input contains the information that is used to construct the forcefield for the molecule types in the system. The choice of inpstyle determines which other variables are required to describe the molecule. Click on the appropriate link for each inpstyle to learn about the remaining variables that are required for each case.

input_style (character*50)

- 'explicit' : Explicit declaration of all terms (formerly **inpstyle** 0)
- 'polypeptide builder' : Polypeptide builder (formerly **inpstyle** 1)
- 'basic connectivity map' : Basic Atom-based connectivity map (formerly **inpstyle** 2)
- 'nucleic acid builder' : Nucleic acid builder (formerly **inpstyle** 3)
- 'nanotube builder' : Nanotube builder (formerly **inpstyle** 4)
- 'atomic' Atomic only: typically for use with external quantum energy calcaultions (formerly **inpstyle** 5)
- 'advanced connectivity map' : Advanced Atom-based connectivity map

Return to the main towhee web page

Send comments to: Marcus G. Martin

Last updated: October 1, 2008

MCCCS Towhee Configurational-bias Monte Carlo

MCCCS Towhee (Configurational-bias Monte Carlo)

Overview

This section gives a basic overview of the Configurational-bias Monte Carlo (CBMC) algorithm that is implemented into Towhee. CBMC algorithm development remains a major research activity, and a particular favorite of at least one of the Towhee developers. For more information about the algorithms implemented into Towhee the best two places to look are the references for each of the moves, and the code itself. The Martin and Siepmann 1999 paper is a good place to start learning about CBMC as it is implemented into Towhee.

Early CBMC algorithms

CBMC was developed on lattice by J. Ilja Siepmann 1990 as a method for sampling chain molecules in a simple model of a monolayer. A variety of researchers (Siepmann and Frenkel 1992, Frenkel *et al.* 1992, Laso *et al.* 1992, and Siepmann and McDonald 1992) brought the method to continuous space in 1992. This version of the CBMC algorithm worked well for the linear chain molecules studied in the mid-nineties and was combined with the Gibbs ensemble to compute some of the first vapor-liquid coexistence curves for chain molecules (such as united-atom *n*-alkanes). The basic concept is that molecules are grown atom by atom into a dense fluid in such a way that the local space for each new atom is sampled and the lower energy positions are more likely to be chosen to continue the growth of the molecule. This accumulates a bias that is then removed in the acceptance rule. The net effect is a large increase in the acceptance rate for insertions of polyatomic molecules into liquids. At this point intramolecular interactions (vibrations, bending angles, dihedrals etc.) were very simple and handled with a Boltzmann rejection scheme.

Dual-cutoff CBMC

In 1998 Vlugt *et al.* 1998 developed a cost-saving version of the CBMC algorithm. Instead of using the full nonbonded cutoff during the CBMC move, a shorter range cutoff is used and this makes the computation less expensive in dense systems. The full potential is then computed for the final structure and the energy difference between the true potential, and the one used to generate the growth trial, is incorporated into the acceptance rule to remove this bias. In Towhee this algorithm is implemented using the **rcutin** variable for the inner cutoff used during the growth and **rcut** for the cutoff used to computed the "true" energy of the system. Proper setting of **rcutin** can decrease the simulation time by a factor of two. Note that the chemical potential computed using dual-cutoff CBMC has not been proven to be correct, and empirical evidence suggests that it is not correct in certain cases.

Coupled-Decoupled CBMC 'Martin and Siepmann 1999' formulation

Work on branched alkane adsorption in silicalite by Vlugt *et al.* 1999 revealed a flaw in the Boltzmann rejection technique if a molecule contains any atom that is bonded to three or more other atoms. In addition, this method became very slow for molecules with atoms bonded to 4 or more other atoms. One of the methods developed to resolve this problem was the coupled-decoupled CBMC algorithm of Martin and Siepmann 1999. The intramolecular terms were now also generated using a biasing procedure with appropriate corrections in the acceptance rule. Bond lengths were still rigid in that paper, although the algorithm was later generalized to include decoupled flexible bond lengths and class 2 force field term by Martin and Thompson 2004. The flexible bond angles were decoupled from the torsions, which were coupled to the nonbonded terms. What this means, is the bond angles are selected based solely on the bond angle energies and the phase space terms and then those angles are used in all subsequent selections (torsion and nonbond). Thus, the bond angle selection is decoupled from the other selections. In contrast, for each nonbond trial a full selection is done to generate torsional angles so these two selections are coupled.

Fixed Endpoint CBMC

Cyclic molecules are substantially more difficult to grow using CBMC because their conformational space is severely limited by the constraints of having cyclic portions of the molecule. Attempting to grow a cyclic molecule using standard CBMC methods, and just hoping it closes itself up properly, has an acceptance rate that is nearly zero. It is generally believed that an additional biasing is required during the growth procedure in order to nudge the growth into positions that will result in reasonable ring closures. There are a variety of biasing procedures in the literature. One that is notable, but not currently implemented into Towhee, is the self-adapting fixed-endpoint (SAFE) CBMC algorithm of Wick and Siepmann 2000. The problem with SAFE-CBMC is it can use a large amount of memory in order to keep track of all of the adapting fixed-endpoint biasing functions. The version implemented into Towhee is currently unpublished, but uses some analytical biasing functions based upon a crude, but consistent, transformation of the distance between growth atoms and target ring atoms, into a bias function based loosely upon dihedral, bending, and vibrational energies. Considerable research is still needed in this area to determine optimal biasing strategies. The algorithm implemented into Towhee was first used, but not satisfactorily described, in Martin and Thompson 2004.

Arbitrary Trial Distribution CBMC

Traditionally, the trials for things like bond lengths, bending angles, and torsional angles are generated according to a distribution that would be the true distribution if there were no potential energy terms. The trials are then accepted or rejected based upon factors related to the potential energy terms. While this split into "entropic" and "energetic" terms is convenient, it is not necessarily the best way to handle the trial generation. Martin and Biddy 2005 used a new method where the bond lengths and bending angles are generated according to a Gaussian distribution and then this is corrected in the acceptance rules. There is nothing special about using a Gaussian and any arbitrary trial distribution can be used to generate the trials so long as it is strictly positive throughout the appropriate ideal range. The strictly positive requirement comes from the acceptance rule which divides out the arbitrary trial distribution in order to remove this bias (division by zero is not a good idea). Several options for the arbitrary trial distributions are implemented into Towhee and this is a subject of continuing research (and hopefully a few journal articles describing the method).

Coupled-Decoupled CBMC 'Coupled to pre-nonbond' formulation

This is an experimental new algorithm that is not yet published. With the invention of the arbitrary trial distribution method it is now possible to get good acceptance rates using only a single trial for things like bond lengths and bending angles, instead of the normal 100 to 1000 required when using the ideal trial distribution method. This enables a rethinking of the coupled-decoupled algorithm as the motivation for decoupling terms was to keep the expense down for the terms that occur early in the growth step, like vibrations and bending angles. Now that these steps are considerably less expensive it makes sense to explore various strategies to try and improve the acceptance rate for challenging molecular geometries (such as strongly branched and cyclic molecules). The 'Coupled to pre-nonbond' formulation implemented into Towhee adds a new selection process in between the dihedral selection and the nonbond selection (a pre-nonbond selection). Bond lengths, bending angles, and dihedral angles are all decoupled from each other, but coupled to the pre-nonbond selection. Preliminary work suggests this can improve the acceptance rate for cyclic molecules. Research in this area is extremely active right now and this implementation is currently in ongoing testing so if you wish to use it be sure to check the website frequently for updates as the debugging process continues.

Return to the main towhee web page

Send comments to: Marcus G. Martin
Last updated: September 12, 2005

MCCCS Towhee (towhee_ff)

Overview

This section described the format that is used to specify Towhee force field files (towhee_ff) for the current version of those file (currently force field version 14, which began in code version 5.1.0). This documentation was last updated for version 5.1.0. Files for many commonly used force fields are provided with the download package and if you are using one of those you do not need to know the details of this section. However, this feature also allows the user to utilize their own force field parameters without having to modify the code themselves. The formatting for this file is extremely sensitive and the user is encouraged to look at the examples in the ForceFields directory. The names of the variables must be listed precisely as shown in this file in order to facilitate error checking of the force field file.

Differences from previous versions

- Version 13: did not have the **Vibration Order** variable for each vibration type, the **Angle Order** variable for each angle type, the **Torsion Order** variable for each torsion type, or the **Bond Increment Order** for each bond increment type. It also included the deprecated **Bond-Angle Strings** variables for certain class 2 angle potentials.
- Version 12: did not have the **Polarizability** variable for each nonbonded atom type and had a typo in the Number of One-Five types label.
- Version 11: included a coefficient for the 'Nonbonded Potential only' Torsion style that was not used for anything.
- Version 10: included a third term for the 'Simple Harmonic Potential' Torsion style that was not used.
- Version 9: had a slightly different potential for the #9 bending angle style.
- Version 8: did not allow the embedding density to be dependent upon both atoms in the pair.
- Version 7: did not have any One-Five interaction information.
- Version 6: had a the **Cross Term Logical** instead of specifying the acceptable classical_mixrule options. Also did not contain the bond increment information.
- Version 5: used the old format of specifying the potential form as an integer instead of as a text string
- Version 4: had a completely separate file format for the Embedded Atom Method potential. This has now been incorporated into a single version. The nbcoeffs for the implicit solvation potential used a different format in this version and is no longer compatable with current versions.
- Version 3: allowed the specification of multiple atoms with the same nonbond parameters and did not utilize the vibration, angle, and torsion name definations. Version 3 and earlier are no longer valid inputs into Towhee.
- Version 2: used to allow the specification of multiple force fields for a single set of parameters. This feature was made obsolete by other changes in the code and was removed in version 3.
- Version 1: did not have the **Cross Term Logical** variable. Instead it was assumed from the **classical_potential**.

Variable explanations for towhee_ff The variable names must all be written exactly as follows. All entries provide the variable name of the descriptive character string, the FORTRAN formatting for the descriptive character string, and the FORTRAN formatting for the actual variable entries.

'towhee_ff Version' (a17) [integer]
- The Version number of the force field file. This documentation is accurate for version 3.

'Number of Nonbonded Types' (a25) [integer]
- The number of nonbonded atom types in the force field file.

'Potential Type' (a14) [character string a30]
- The nonbonded potential type. See the classical_potential documentation for information about how to set this variable. Only the **classical_potential** is needed here and not all of the variables associated with the **classical_potential**.

'Classical Mixrule' (a17) [character string a30]
- The classical mixing rule that works in combination with this forcefield. Generally this must match exactly with the **classical_mixrule** variable (set in towhee_input). However, certain special combination cases are also allowed. Currently the only combination case is 'LB or Geometric' which matches with a **classical_mixrule** of 'Lorentz-Berthelot' or 'Geometric'.

This section is repeated once for each of the Number of Nonbonded Types.

'Atom Type Number' (a16) [integer]

- The number coresponding to the atom type. These numbers must run from 1 to the Number of Nonbonded Types.

The following set of variables are repeated once for each atom type pair represented here. If the '**Classical Mixrule**' is 'Explicit' then there are multiple listings (interaction of this **Atom Type Number** with itself and with all other atoms that have a larger Atom Type Number). For any other value of the '**Classical Mixrule**' there is only a single listing (interaction of this Atom Type Number with itself).

'eam_pair_style' (a14) [character string a20]

This variable is only listed if **Potential Type** is set to 6 (Embedded Atom Method). This is the style of the EAM pair interaction for this combination of atom types. Currently supported options are listed here.

- 'Ackland 3-part': The pair potential is a complicated 3 part function as described in Mendelev *et al.* 2003.

```
if r < nb(1) then
    x = r/nb(4)
    $u_{pair}$ = ( 0.1818 * exp[-3.2 * x] + 0.5099 * exp[-0.9423 * x] + 0.2802 * exp[-0.4029 * x] + 0.02817 * exp[-0.2016 * x] ) * nb(3) / r
elseif r < nb(2) then
    x = nb(5) + nb(6) * r + nb(7) * $r^2$ + nb(8) * $r^3$
    $u_{pair}$ = nb(9) * exp[x]
else
    $u_{pair}$ = $Sum_{i = 1,table\_npair}$ table_pair(2,i) * [table_pair(1,i) - r]$^3$ * Heaviside[table_pair(1,i)-r]
```

- 'Ackland Power': The pair potential is a complicated 3 part function as described in Ackland *et al.* 2004.

```
if r < nb(1) then
    $u_{pair}$ = ( nbcoeff(4) * exp[nbcoeff(5) * r] + nbcoeff(6) * exp[nbcoeff(7) * r] + nbcoeff(8) * exp[nbcoeff(9) * r] + nbcoeff(10) * exp[nbcoeff(11) * r] ) * nb(3) / r
elseif r < nb(2) then
    x = nb(12) + nb(13) * r + nb(14) * $r^2$ + nb(15) * $r^3$
    $u_{pair}$ = nb(16) * exp[x]
else
    $u_{pair}$ = $Sum_{i = 1,table\_npair}$ table_pair(2,i) * [table_pair(1,i) - r]$^{i+3}$ * Heaviside[table_pair(1,i)-r]
```

- 'exponential': The pair potential is described by an exponential function.
  $u_{pair} = nb_1\, e^{[nb_2\, r_{ij}]}$.
- 'morse': The pair potential is described by the Morse function.
  $u_{pair} = nb_1 * \{ \exp[-2\, nb_2\, (r - nb_3)] - 2 \exp[-nb_2\, (r-nb_3)]\}$
- 'table': This is the traditional way of describing EAM potential information and presents a table of data that is interpolated to determine the energy for any distance.

'table_pair' (a11) [integer, integer, integer]

This variable is only listed for tabular potentials. This occurs in the following cases

- **classical_potential** 'Embedded Atom Method' and **eam_pair_style** 'Ackland 3-part'
- **classical_potential** 'Embedded Atom Method' and **eam_pair_style** 'Ackland Power'
- **classical_potential** 'Embedded Atom Method' and **eam_pair_style** 'table'
- **classical_potential** 'Multiwell'.
- **classical_potential** 'Repulsive Multiwell'.
- **classical_potential** 'Tabulated Pair'

- Three values on a single line. First atom type, second atom type, and the number of tabular data lines of **table_pair_data**

'table_pair_data' (a15) [double precision, double precision]

- The tabular values used to determine the pair potential. This is only listed for tabular potentials. There are two entries per line (table_pair(1,n) and table_pair(2,n)) and the number of lines is equal to the value set in **table_pair**

'Nonbond Coefficients' (a20) [double precision]

- These values are listed for any non-tabular potential and they contain all of the nonbonded coefficients. These are listed with one coefficient per line. The number of nonbonded coefficients depends upon the **classical_potential**. The native units of the code are used (Kelvin for energy, Angstrom for distances). If the '**Classical Mixrule**' is 'Explicit' then the cross terms must be explicitly declared for all interactions of this atom type with any atom type that has a larger Atom Type Number. Otherwise the nonbonded coefficients are listed only for self-interaction and the mixing rule will set the cross terms.

End of the section that is repeated based on the Number of Atom Type Pairs.

The following section is only included for **classical_potential** 'Embedded Atom Method'. The density contributed to all of the other atom types is listed here for the current atom type.

'eam_dens' (a8) [integer, integer, integer]

- Three values on a single line. Atom type the density is contributed to, atom type the density is contributed from, and number of lines of density coefficients

'eam_dens_style' (a14) [character string a20]

The style of the EAM density interaction for this atom type. There are several options currently supported by Towhee.

- 'Ackland cubic sum'

  rho = Sum$_{i = 1,eam\_ndens}$ eam_dens(2,i) * [eam_dens(1,i) - r]^3 * Heaviside[eam_dens(1,i)-r]

- 'exponential': The density is described by an exponential function.

  $rho = A e^{B r_{ij}}$.

- 'table': This is the traditional way of describing EAM density information and presents a table of data that is interpolated to determine the density for any distance.

'eam_dens_data' (a14) [double precision, double precision]

The values used to determine the embedding density. The meanings are different depending on the setting of **eam_dens_style** above. In all cases there are a number of lines equal to the value set in **eam_dens**

- 'table': Each line contains two values. The first is the distance (Angstroms) and the second is the corresponding embedding density (arbitrary units).
- 'exponential': The embedding density is described by an exponential function. The format is a bit clumsy as the first line contains a dummy value and then the *A* prefactor (arbitrary units). The second line contains a dummy value and then the *B* exponential factor (inverse Angstroms).
- 'Ackland cubic sum': Each line contains two values. The first is a distance (Angstroms) and the second is a density prefactor (arbitrary units).

'eam_embed' (a9) [integer, integer]

- Only listed for **classical_potential** 'Embedded Atom Method'. Two values on a single line. Atom type and number of lines of embedding function coefficients.

'eam_embed_style' (a15) [character string a20]

Only listed for **classical_potential** 'Embedded Atom Method'. The style of the EAM embedding function for this atom type. Currently supported options are listed below.

- 'table': This is the traditional way of describing EAM embedding function information and presents a table of data that is interpolated to determine the embedding function energy for any density.
- 'Ackland 2-term': $u_{embed}(rho)$ = eam_embed(1,1) * [- $rho^{0.5}$ + eam_embed(2,1) * rho^2]

'eam_embed_data' (a15) [double precision, double precision]

Only listed for **classical_potential** 'Embedded Atom Method'. The values used to determine the embedding function energy. The meanings are different depending on the setting of **eam_embed_style** above. In all cases there are a number of lines equal to the value set in **eam_embed**

- 'table': Each line contains two values. The first is the density (arbitrary units) and the second is the corresponding embedding function energy (Kelvin/k_b).

  - 'Ackland 2-term': A single line that contains a prefactor (eam_embed(1,1)) and a square factor (eam_embed(2,1) as described above.

End of the special 'Embedded Atom Method' section.

'Mass' (a4) [double precision]
- The atomic mass of this element. Units are grams/mol.

'Element' (a7) [character string a2]
- The elemental name of this Atom Type.

'Bond Pattern' (a12) [character string a5]
- The bond pattern for this Atom Type. The bond pattern is used with configurational-bias moves that utilize a non-uniform generation of the bond angles and dihedrals. See the **configurational-bias manual** for more information. Set to the word 'null' if you do not want to use this feature.

'Base Charge' (a12) [double precision]
- The base charge on this atom type for use with the 'bond increment' method of automatically assigning charges. This value is zero for most atom types, but can take on different values for ionic systems.

'Polarizability' (a14) [double precision]
- The polarizability of each atom type for use with some of the classical mixing rules. Expressed in units of cubic Angstroms.

'Force Field Name' (a16) [character string a10]
- The name of the Force Field

'Atom Names' (a10) [character string a10 on four lines]
- Each Atom Type has a nonbonded name, a bonded name, an angle name, and a torsion name. These are listed in that order, one per line. These names are used in combination with the assemble features in Towhee to make it easier to build molecule structures in the towhee_input file.

End of the section that is repeated based on the Number of Nonbonded Types.

'Number of Bonded Terms' (a22) [integer]
- The number of different Bonded terms in the force field file.

This section is repeated once for each of the Number of Bonded Terms

'Bond Type Number' (a16) [integer]
- The number coresponding to the bond type. These numbers must run from 1 to the Number of Bonded Terms.

'Bond Style' (a10) [integer]
- The number coresponding to the bond style. Below is a list of Bond Styles, their potential form, and the required vibcoeffs. The equilibrium bond length is always listed in vibcoeff(0).
  - 1: Fixed Bond Length

    U_{bond} = Infinity, if (length - vibcoeff(0)) / vibcoeff(0) > 0.01

    U_{bond} = 0 otherwise.
  - 2: Standard Harmonic

    U_{bond} = vibcoeff(1)*[length - vibcoeff(0) $]^2$
  - 3: Gromos Quartic

    U_{bond} = vibcoeff(1)*[length^2 - vibcoeff(0)^2 $]^2$
  - 4: Nonlinear

    distance = [length - vibcoeff(0) $]^2$

    U_{bond} = [vibcoeff(1)*distance] / [vibcoeff(2) - distance]
  - 5: MM2 Quadradic and Triplet

    distance = [length - vibcoeff(0)]

    U_{bond} = vibcoeff(1)*distance^2 * [1.0 - 2.0*distance]
  - 6: Compass Quartic

distance = [length - vibcoeff(0)]

U_{bond} = vibcoeff(1)*distance^2 + vibcoeff(2)*distance^3 + vibcoeff(3)*distance^4

- 7: Nonbonded interactions used as the Bonding potential
  This has no vibcoeffs as it uses the nonbond van der Waals and coulombic potentials instead.
- 8: No interaction
  Zero energy regardless of interatomic separation.
- 9: Morse
  distance = [length - vibcoeff(0)]
  U_{bond} = vibcoeff(1)*[$e^{vibcoeff(2)*distance}$ - 1]^2
- 10: Infinite Square Well
  vibcoeff(0) should be set to the arithmetic mean of vibcoeff(1) and vibcoeff(2) for use with certain configurational-bias options despite not explicitly appearing in the energy calculation.
  U_{bond} = infinity if length < vibcoeff(1)
  U_{bond} = vibcoeff(3) if vibcoeff(1) <= length <= vibcoeff(2)
  U_{bond} = infinity if vibcoeff(2) < length
- 11: Standard Harmonic plus nonbonded interactions
  U_{bond} = vibcoeff(1)*[length - vibcoeff(0)]^2 + $U_{nonbond}(r_{ij})$ + $U_{electrostatic}(r_{ij})$

'Bond Coefficients' (a17) [double precision]
- The values of the vibcoeffs listed in order (starting with 0) with a single coefficient per line.

'Vibration Order' (a15) [character string a10]
- This is used to distinguish between bond types for atoms that can make more than one kind of bond (for example, sp^2 carbons bond to each other with either single or double bonds). Commonly used options are listed in the inpstyle 6 manual.

'Force Field Name' (a16) [character string a10]
- The name of the Force Field

'Number of Atoms with Same Parameters' (a38) [integer]
- The number of pairs of atoms in this force field which use the same bond parameters.

'Atom Names' (a10) [pairs of character strings a10,1x,a10]
- All of the pairs of Atom Names for this Bond Type. These are listed with one pair of atom names per line.

End of the section repeated based on the Number of Bonded Terms

---

'Number of Angle Terms' (a21) [integer]
- The number of different Angle terms in the force field file.

This section is repeated based on the Number of Angle Terms

'Angle Type Number' (a17) [integer]
- The number coresponding to the angle type. These numbers must run from 1 to the Number of Angle Terms.

'Angle Style' (a11) [integer]
- The number coresponding to the angle style. Below is a list of Angle Styles,their potential form, and the bencoeffs. All of the angles are listed using degrees in the force field file, and are converted into radians for use in the code. Thus, all of the force constants need to be appropriate for use with radians, while the angles are listed in degrees. The equilibrium angle is always listed in bencoeff(0) and only this value is converted from degrees to radians.
  - 0: Rigid Angle
    $U_{angle}(\theta)$ = Infinity, if $|(\theta$ - bencoeff(0))$|$ > bencoeff(1)
    $U_{angle}(\theta)$ = 0 otherwise.
  - 1: Standard Harmonic
    $U_{angle}(\theta)$ = bencoeff(1)*[θ - bencoeff(0)]^2

- 2: DREIDING 1 + Cos(θ)
  $U_{angle}(\theta)$ = bencoeff(1)*[1.0 + Cos(θ)]
- 3: Harmonic Cosine
  $U_{angle}(\theta)$ = bencoeff(1)*[Cos(θ) - Cos(bencoeff(0)) $]^2$
- 4: Compass Quartic Angle with autodetection
  The Bond-Angle and Bond-Bond (1-2) Class 2 cross terms are incorporated into this angle and that makes the potential assymetric with respect to the atom order. This uses the Compass atom type definitions to decide the order of the terms. Then, it finds the appropriate bond terms for use in the bond-angle and bond-bond cross terms. Two logicals control whether there are any Bond-Angle or Bond-Bond cross terms. Definitions: r_{ij} is the distance between the first and second atoms in the angle, r_{jk} is the distance between the second and third atoms of the angle, and the appropriate vibcoeff(0) values are determined automatically.
  differ = θ - bencoeff(0)
  $U_{angle}(\theta)$ = bencoeff(1)*differ^2 + bencoeff(2)*differ^3 + bencoeff(3)*differ^4 + $U_{bond\text{-}angle}(r_{ij},r_{jk},\theta)$ + $U_{bond\text{-}bond}(r_{ij},r_{jk})$
  if **Bond-Angle Logical** is true then
  $U_{bond\text{-}angle}(\theta,r_{ij},r_{jk})$ = differ*[bencoeff(4)*(r_{ij} - vibcoeff(0)$_{ij}$) +bencoeff(5)*(r_{jk} - vibcoeff(1)$_{jk}$)]
  otherwise
  $U_{bond\text{-}angle}(r_{ij},r_{jk},\theta) = 0$
  if **Bond-Bond Logical** is true then
  $U_{bond\text{-}bond}(r_{ij},r_{jk})$ = bencoeff(6)*[(r_{ij} - vibcoeff(0)$_{ij}$) *(r_{kj} - vibcoeff(0)$_{jk}$)]
  otherwise
  $U_{bond\text{-}bond}(r_{ij},r_{jk}) = 0$
- 5: Charmm Harmonic with Urey-Bradley
  $U_{angle}(\theta,d_{ik})$ = bencoeff(1)*[θ - bencoeff(0) $]^2$ + bencoeff(3)*[d_{ik} - bencoeff(2) $]^2$
  where d_{ik} is the distance between the first and third atoms of the angle.
- 6: Nonbonded Terms only between the 1-3 atoms of the angle
  The angle energies are actually the nonbonded van der Waals plus the nonbonded coulomb terms. Towhee requires an angle term between all atoms connected by two bonds, so this term is used if you actually want just a nonbonded interaction (for example, with a freely jointed chain). There are no bencoeffs for this type.
- 7: Nonbonded Terms between the 1-3 atoms plus a harmonic angle
  The angle energies are the nonbonded van der Waals, the nonbonded coulomb terms, and the following.
  $U_{angle}(\theta)$ = bencoeff(1)*[θ - bencoeff(0)$]^2$
- 8: Compass Quartic Angle with explicit ordering of terms
  The Bond-Angle and Bond-Bond (1-2) Class 2 cross terms are incorporated into this angle and that makes the potential assymetric with respect to the atom order. This version uses a positive or negative convenction on the type numbers (assigned in towhee_input) to tell the code whether to follow the order as presented here (positive) or to use the inverse order (negative). Two logicals control whether there are any Bond-Angle or Bond-Bond cross terms.
  differ = angle-bencoeff(0)
  $U_{angle}(\theta)$ = bencoeff(1)*differ^2 + bencoeff(2)*differ^3 + bencoeff(3)*differ^4
  if there are Bond-Angle cross terms then
  $U_{bond\text{-}angle}(\theta,d_{ij},d_{jk})$ = [θ - bencoeff(0)] * {bencoeff(4)*[d_{ij} - bencoeff(5)] + bencoeff(6)*[d_{jk} - bencoeff(7)]}
  $U_{bond\text{-}bond}(d_{ij},d_{jk})$ = bencoeff(8) * [d_{ij} - bencoeff(9)] * [d_{jk} - bencoeff(10)]

- 9: Fourier Expansion with constant plus single term
  This was originally implemented for use with the UFF forcefield. The prefactor is computed in a much more elaborate manner than is usual for angle force constants and only works in combination with the UFF 12-6 classical_potential. This prefactor is computed for each angle in the simulation during the structure check at the start of a simulation. Note: this potential is not well designed as it also has a minimum for a nonphysical θ value of zero.
  θ = bencoeff(0)
  r_{ik} = [vibcoeff(ij,0)^2*vibcoeff(jk,0)^2 - 2*vibcoeff(ij,0)*vibcoeff(jk,0)*Cos(θ)]$^{1/2}$
  benprefact(ijk) = kcaltok*664.12*nbcoeff(10,i)*nbcoeff(10,k)/($r_{ik}{}^5$) * {
  vibcoeff(ij,0)*vibcoeff(jk,0)*[1 - Cos^2(θ)] - vibcoeff(ik,0)^2*Cos(θ) } / bencoeff(1)^2
  U_{angle}(θ) = benprefact(ijk) * [1 + bencoeff(2) * Cos(bencoeff(1)*θ)]
- 10: Three Term Fourier Expansion
  This was originally implemented for use with the UFF forcefield. The prefactor is computed in a much more elaborate manner than is usual for angle force constants and only works in combination with the UFF 12-6 classical_potential. This prefactor is computed for each angle during the structure check at the start of a simulation. Note: this prefactor is nearly identical to the one for anglestyle 9, with the notable exception of not dividing by the bencoeff(1)^2 term.
  θ_0 = bencoeff(0)
  r_{ik} = [vibcoeff(ij,0)^2*vibcoeff(jk,0)^2 - 2*vibcoeff(ij,0)*vibcoeff(jk,0)*Cos(θ_0)]$^{1/2}$
  benprefact(ijk) = kcaltok*664.12*nbcoeff(10,i)*nbcoeff(10,k)/($r_{ik}{}^5$) * {
  vibcoeff(ij,0)*vibcoeff(jk,0)*[1 - Cos^2(θ_0)] - vibcoeff(ik,0)^2*Cos(θ_0) }
  U_{angle}(θ) = benprefact(ijk) * [bencoeff(1) + bencoeff(2)*Cos(θ) + bencoeff(3)*Cos(2θ)]
- 11: No interaction
  The angle energy is zero regardless of the distance or angle between the atoms.
- 12: Sixth Power Angle with Autodetection of Angle-Bond Terms
  The Bond-Angle term is incorporated into this angle and that makes the potential assymetric with respect to the atom order. This version uses the first letter of the angle atom type definitions to decide the order of the terms. Then, it finds the appropriate bond terms for use in the bond-angle term.
  differ = θ - bencoeff(0)
  U_{angle}(θ) = bencoeff(1)*differ^2 * (1 + bencoeff(2)*differ^4)
  if there are Bond-Angle cross terms then
  $U_{bond\text{-}angle}$(θ) = bencoeff(3) * [θ_{ijk} - bencoeff(0)] * [(r_{ij} - Equilibrium bond length$_{ij}$) + (r_{jk} - Equilibrium bond length$_{jk}$)]
- 13: Infinite Square Well Angle
  The angle terms is a 1,3 distance based check with bounds.
  U_{angle}(θ) = infinity if distance$_{1\text{-}3}$ < bencoeff(1)
  U_{angle}(θ) = bencoeff(3) if bencoeff(1) <= distance$_{1\text{-}3}$ <= bencoeff(2)
  U_{angle}(θ) = infinity if bencoeff(2) < distance$_{1\text{-}3}$
- 14: Multiple Allowed Rigid Angles
  U_{angle}(θ) = Infinity, if | (θ - bencoeff(0)) | > bencoeff(2) AND | (θ - bencoeff(1)) | > bencoeff(2).
  U_{angle} = 0 otherwise.
- 15: MMFF Cubic with bond-angle cross terms
  differ = θ - bencoeff(0)
  U_{angle}(θ) = bencoeff(1)*differ^2 + bencoeff(2)*differ^3 + $U_{bond\text{-}angle}$(r_{ij},r_{jk},θ)
  if **Bond-Angle Logical** is true then

$U_{bond-angle}(r_{ij},r_{jk},\theta)$ = differ*[bencoeff(3)*(r_{ij} - vibcoeff$(0)_{ij}$) +bencoeff(4)*(r_{jk} - vibcoeff$(0)_{jk}$)]

otherwise

$U_{bond-angle}(r_{ij},r_{jk},\theta) = 0$

- 16: Harmonic Cosine plus 1-3 nonbonded interactions

  $U_{angle}(\theta)$ = bencoeff(1)*[Cos(θ) - Cos(bencoeff(0)) $]^2$ + $U_{nonbond}(r_{ik})$ + $U_{electrostatic}(r_{ik})$

'Bond-Angle Logical' (a18) [logical]

- This entry is only present for the Compass angle styles (4 and 8). It is true if there are bond-angle terms and false otherwise.

'Bond-Angle Coefficients' (a23) [double precision]

- This entry is only present for the Compass angle styles (4 and 8), and only if the Bond-Angle Logical is true. This lists the Bond-Angle coefficients (those appearing in $U_{bond-angle}$) one per line, in order starting from the lowest index coefficient.

'Bond-Bond Logical' (a17) [logical]

- This entry is only present for the Compass angle styles (4 and 8). This is true if there are bond-bond terms with this angle type.

'Bond-Bond Coefficients' (a22) [double precision]

- This entry is only present for the Compass angle styles (4 and 8), and only if the Bond-Bond Logical is true. This lists the Bond-Bond coefficients (those appearing in $U_{bond-bond}$) one per line, in order starting from the lowest index coefficient.

'Angle Coefficients' (a18) [double precision]

- The values of the bencoeffs listed in order (starting with 0) with a single coefficient per line. Note, that those bencoeffs that appear in the bond-angle or bond-bond cross terms are not listed here.

'Angle Order' (a11) [character string a15]

- This is used to distinguish between angle types that contain the same sets of angles, but in different environments. The environment is automatically detected according to the appropriate rules for the given forcefield. Required for all forcefields, but currently only used for MMFF94

'Force Field Name' (a16) [character string a10]

- The name of the Force Field

'Number of Atoms with Same Parameters' (a38) [integer]

- The number of triplets of atoms in this force field which use the same angle parameters.

'Atom Names' (a10) [character string triplets a10,1x,a10,1x,a10]

- All of the triplets of Atom Names for this Angle Type. These are listed with one triplet of atom names per line.

End of the section repeated based on the Number of Angle Terms

---

'Number of Torsion Terms' (a23) [integer]

- The number of different regular torsion terms in the force field file. Regular torsions are defined for atoms that are connected through exactly three consecutive bonds.

This is repeated based on the Number of Torsion Terms

'Torsion Type Number' (a19) [integer]

- The number coresponding to the torsion type. These numbers must run from 1 to the Number of Torsion Terms.

'Torsion Style' (a13) [integer]

- The number coresponding to the torsion style. Below is a list of Torsion Styles,their potential form, and the torcoeffs. The torsional potentials have quite a variety of forms (as shown below). Any energy terms need to use the standard code units of Kelvins. All torsion angles are computed in radians within Towhee, so any additive terms with the torsion angles should also be listed in radians. The torsional angle is listed as f in the equations below. Please note that different authors define the torsional zero differently. Towhee uses the convention that a perfect cis bond has a torsional angle of 0.

- 1: Simple Harmonic Potential
  $U_{torsion}$(f) = torcoeff(0) * [f - torcoeff(1)]^2
- 2: OPLS style Cosine Series
  $U_{torsion}$(f) = torcoeff(1) * [1 + Cos(f)] + torcoeff(2) * [1 - Cos(2f)] + torcoeff(3) * [1 + Cos(3f)]
- 3: Gromos/Charmm/Amber style Cosine Series
  $U_{torsion}$(f) = $Sum^{i=1,ntorloop}$ torcoeff(3*(i-1)+1) * [1 + Cos(torcoeff(3*(i-1)+2)f - torcoeff(3*(i-1)+3))]
- 4: Gromos/Charmm/Amber style Cosine Series plus a harmonic term that Charmm traditionally calls an improper term despite the fact that the bonding pattern is that of a regular torsion.
  $U_{torsion}$(f) = [$Sum^{i=1,ntorloop}$ torcoeff(3*(i-1)+1) * [1 + Cos(torcoeff(3*(i-1)+2)f - torcoeff(3*(i-1)+3))]]
  + torcoeff(3*ntorloop+1) * [f - torcoeff(3*ntorloop+2)]^2
- 5: Compass Cosine series with cross terms and with autodetection.
  The bond and angle terms that appear in the cross terms use the appropriate vibcoeff and bencoeff parameters for the bond or angle in question. These are determined using the Compass naming conventions.
  $U_{torsion}$(f) = torcoeff(0) * [1 - Cos(f)] + torcoeff(1) * [1 - Cos(2f)] + torcoeff(2) * [1 - Cos(3f)]
  $U_{bond\text{-}torsion}$ = (bond term 1-2) * [torcoeff(3) * Cos(f) + torcoeff(4) * Cos(2f) + torcoeff(5) * Cos(3f)]
  + (bond term 2-3) * [torcoeff(6) * Cos(f) + torcoeff(7) * Cos(2f) + torcoeff(8) * Cos(3f)]
  + (bond term 3-4) * [torcoeff(9) * Cos(f) + torcoeff(10) * Cos(2f) + torcoeff(11) * Cos(3f)]
  $U_{angle\text{-}torsion}$ = (angle term 1-2-3) * [torcoeff(12) * Cos(f) + torcoeff(13) * Cos(2f) + torcoeff(14) * Cos(3f)]
  + (angle term 2-3-4) * [torcoeff(15) * Cos(f) + torcoeff(16) * Cos(2f) + torcoeff(17) * Cos(3f)]
  $U_{angle\text{-}angle\text{-}torsion}$ = torcoeff(18) * (angle term 1-2-3) * (angle term 2-3-4)
  $U_{bond\text{-}bond}$ = torcoeff(19) * (bond term 1-2) * (bond term 3-4)
- 6: Compass Cosine series without cross terms and with autodetection.
  $U_{torsion}$(f) = torcoeff(0) * [1 - Cos(f)] + torcoeff(1) * [1 - Cos(2f)] + torcoeff(2) * [1 - Cos(3f)]
- 7: TraPPE simple Cosine function.
  Note that unlike most force fields, TraPPE uses the convention that a trans bond is 0 f. That is the reason for the shift of p in this potential as Towhee uses the zero cis convention.
  $U_{torsion}$(f) = torcoeff(0) * [1 - Cos(2*[f - p] + torcoeff(1))]
- 8: Nonbonded Potential only.
  The nonbonded potential for van der Waals and coulombic terms is the only thing used.
- 9: Compass cosine series with cross terms and explicit parameter declaration.
  $U_{torsion}$(f) = torcoeff(0) * [1 - Cos(f - torcoeff(3))] + torcoeff(1) * [1 - Cos(2*(f - torcoeff(4)))] + torcoeff(2) * [1 - Cos(3*(f - torcoeff(5)))]
  $U_{bond\text{-}torsion}$(f) = [length(1-2) - torcoeff(9)] * [torcoeff(6) * Cos(f) + torcoeff(7) * Cos(2f) + torcoeff(8) * Cos(3f)]
  + [length(2-3) - torcoeff(13)] * [torcoeff(10) * Cos(f) + torcoeff(11) * Cos(2f) + torcoeff(12) * Cos(3f)]
  + [length(3-4) - torcoeff(17)] * [torcoeff(14) * Cos(f) + torcoeff(15) * Cos(2f) + torcoeff(16) * Cos(3f)]
  $U_{angle\text{-}torsion}$(f) = [angle(1-2-3) - torcoeff(24)] * [torcoeff(18) * Cos(f) + torcoeff(19) *

Cos(2f) + torcoeff(20) * Cos(3f)]
+ [angle(2-3-4) - torcoeff(25)] * [torcoeff(21) * Cos(f) + torcoeff(22) * Cos(2f) + torcoeff(23) * Cos(3f)]
$U_{angle-angle-torsion}$(f) = torcoeff(26) * [angle(1-2-3) - torcoeff(27)] * [angle(2-3-4) - torcoeff(28)] * [Cos(f - torcoeff(3))]
$U_{bond-bond}$ = torcoeff(29) * [length(1-2) - torcoeff(30)] * [length(3-4) - torcoeff(31)]

- 10: Cosine Power Series.
  Note that there are actually ntorloop+1 parameters required as this sum runs from a power of 0 to a power of ntorloop.
  $U_{torsion}$(f) = $\mathrm{Sum}^{i=0,ntorloop}$ torcoeff(i)*$\mathrm{Cos}(f)^{i}$
- 11: Old style OPLS Cosine Series.
  $U_{torsion}$(f) = torcoeff(0) + torcoeff(1)*[1 + Cos(f)] + torcoeff(2)*[1 - Cos(2f)] + torcoeff(3)*[1 + Cos(3f)]
- 12: Sum2003 Cosine Series.
  $U_{torsion}$(f) = $\mathrm{Sum}^{i=0,ntorloop}$ torcoeff(i+1) * [1 - Cos{i*(f - torcoeff(0))}]
- 13: Old OPLS Two term Cosine Series.
  $U_{torsion}$(f) = torcoeff(1)*[1 - Cos(f)] + torcoeff(2)*[1 - Cos(2f)]
- 14: UFF 1 - Cosine divided by Total Number of Torsions
  This potential has the initial force constant divided by the total number of torsions that share the central two atoms. This value is computed on the fly based upon the number of atoms bonded to each of the two central atoms.
  totbond = [invib($atom_2$)-1]*[invib($atom_3$)-1]
  $U_{torsion}$(f) = torcoeff(1) * [1 - torcoeff(2)*Cos(torcoeff(3)f)]/totbond
- 15: DREIDING 1 - Cos(n (f - f_0)) divided by Total Number of Torsions
  This potential has the initial force constant divided by the total number of torsions that share the central two atoms. This value is computed on the fly based upon the number of atoms bonded to each of the two central atoms.
  totbond = [invib($atom_2$)-1]*[invib($atom_3$)-1]
  $U_{torsion}$(f) = torcoeff(1) * [1 - Cos(torcoeff(2)*(f - torcoeff(3)))]/totbond
- 16: 2-fold Cosine
  $U_{torsion}$(f) = torcoeff(1) * [1 - Cos(2f)]
- 17: TraPPE planer cosine series
  $U_{torsion}$(f) = torcoeff(1) * [1 + Cos(f + torcoeff(3))] + torcoeff(2) * [1 - Cos^2(f)]
- 18: Square Well
  $U_{torsion}$ = torcoeff(3): if torcoeff(1) < r_{ik} < torcoeff(2)
  $U_{torsion}$ = infinity : otherwise
- 19: Amber / Number of Torsions
  tottor(jk) = Sum of all torsions that share j and k as the central atoms
  $U_{torsion}$(f) = $\mathrm{Sum}^{i=1,ntorloop}$ torcoeff(3*(i-1)+1) * [1 + Cos(torcoeff(3*(i-1)+2)f - torcoeff(3*(i-1)+3))] / tottor(jk)
- 20: OPLS Fluorocarbon Four Term
  $U_{torsion}$(f) = torcoeff(1)*(1 + Cos(f)) + torcoeff(2)*(1 - Cos(2f)) + torcoeff(3)*(1 + Cos(3f)) + torcoeff(4)*(1 - Cos(4f))
- 21: Multiple Rigid Dihedrals
  $U_{torsion}$(f) = Infinity if for each value of **x** as **x** goes from 1 to **ntorloop**
  | f - torcoeff(**x**) **|** > torcoeff(0)
  $U_{torsion}$(f) = 0 otherwise.
- 22: Fluoroalkane series from Cui *et al.* 1998
  $U_{torsion}$(f) = torcoeff(0) + torcoeff(1)*(1 - Cos(f)) + torcoeff(2)*(1 - Cos(3 f)) +

5 2

torcoeff(3)*(1 - Cos(f)) + torcoeff(4)*exp[-torcoeff(5) f]

'One-Four Nonbond Logical' (a24) [logical]

- True if you wish to add nonbonded van der Waals and coulombic terms into the torsional potential.

'One-Four Coulombic Scaling' (a26) [double precision]

- This entry is only present if the One-Four Nonbond Logical is true. This is the scaling prefactor for the coulombic one-four terms in the torsional potential. A value of 1.0 uses the same coulombic energy as for all of the other nonbonded terms.

'Number of Torsion Loops' (a23) [integer]

- This entry is only present for those potentials that have the ntorloop variable (see above). In those cases, this is the number of terms in the Cosine series.

'Torsion Coefficients' (a20) [double precision]

- The torcoeff variables, listed one per line, starting with the lowest index variable and working upwards.

'Torsion Order' (a13) [character string a15]

- This is used to distinguish between torsion types that contain the same sets of atoms, but in different environments. The environment is automatically detected according to the appropriate rules for the given forcefield. Required for all forcefields, but currently only used for MMFF94

'Force Field Name' (a16) [character string a10]

- The name of the Force Field

'Number of Atoms with Same Parameters' (a38) [integer]

- The number of quartets of atoms in this force field which use the same torsion parameters.

'Atom Names' (a10) [character string quartets a10,1x,a10,1x,a10,1x,a10]

- All of the quartets of Atom Names for this Torsion Type. These are listed with one quartet of atom names per line.

End of the section repeated based on the Number of Torsion Terms

---

'Number of Improper Terms' (a24) [integer]

- The number of different improper torsion terms in the force field file. Improper torsions are defined for three atoms that are all bonded to a single central atom. They are most often used to enforce planarity of sp^2 centers.

This section is repeated based on the Number of Improper Terms

'Improper Type Number' (a20) [integer]

- The number coresponding to the improper type. These numbers must run from 1 to the Number of Improper Terms.

'Improper Form' (a13) [integer]

- The number coresponding to the improper form.
  - 1: Amber / Stereocenter Form.
  - 2: Charmm / Simple out-of-plane form.
  - 3: Average over the three possible Wilson out-of-plane angles form.
  - 4: UFF angle between the IL vector and the IJK plane where I is the central atom bonded to atoms J, K, and L.
  - 5: Vector of the three possible Wilson out-of-plane angles

'Improper Style' (a13) [integer]

- The number coresponding to the improper style. Below is a list of Improper Styles and their impcoeffs. The improper potentials have a variety of forms (as shown below). Any energy terms need to use the standard code units of Kelvins. All improper angles are computed in radians within Towhee, so any additive terms with the improper angles should also be listed in radians. The improper angle is listed as f in the equations below. Please note that different authors define the torsional zero differently. Towhee uses the convention that a perfect cis bond has a torsional angle of 0.
  - 1: Simple Harmonic Potential

    $U_{improper}(f)$ = impcoeff(0) * [f - impcoeff(1)]^2
  - 2: OPLS style cosine series

$U_{improper}$(f) = impcoeff(1) * [1 + Cos(f)] + impcoeff(2) * [1 - Cos(2f)] + impcoeff(3) * [1 + Cos(3f)]

- 3: Potentials for enforcing stereochemistry
  This is to be used only with the stereochemistry improper form. It is used to enforce R/S or D/L stereochemistry centers. The same potential is used in both cases, but the atom ordering is different.
  $U_{improper}$(f) = 0, if 0 < f ;
  = impcoeff(0) + impcoeff(1)*[-f], if -0.5 p < f < 0
  = impcoeff(0) + impcoeff(1)*[p + f], if f < -0.5 p
- 4: Amber cosine
  $U_{improper}$(f) = impcoeff(1) * [1 + Cos(impcoeff(2)f - impcoeff(3))]
- 5: Wilson Harmonic
  $U_{improper}$(f) = impcoeff(1) * [1 + Cos(impcoeff(2)f - impcoeff(3))]
- 6: UFF Cosine Series
  $U_{improper}$(f) = impcoeff(3)*[impcoeff(0) + imcoeff(1)*Cos(f) + impcoeff(2)*Cos(2f)]
- 7: 1 - Cos(f)
  $U_{improper}$(f) = impcoeff(1)*[1 - Cos(f)]
- 8: MMFF Wilson planar enforcer (only works with **Improper Form** 5)
  $U_{improper}$(f) = impcoeff(1)*[$f_j^2 + f_k^2 + f_l^2$]

'Improper Coefficients' (a21) [double precision]

- The impcoeff variables, listed one per line, starting with the lowest index variable and working upwards.

'Force Field Name' (a16) [character string a10]

- The name of the Force Field

'Number of Atoms with Same Parameters' (a38) [integer]

- The number of quartets of atoms in this force field which use the same improper parameters

'Atom Names' (a10) [character string quartets a10,1x,a10,1x,a10,1x,a10]

- All of the quartets of Atom Names for this Improper Type. These are listed with one quartet of atom names per line. The naming conventions for the different force fields are not consistent. If you wish to use this feature you should look in the code for examples.

End of the section repeated based on the Number of Improper Terms

---

'Number of Angle-Angle Terms' (a27) [integer]

- The number of different angle-angle terms in the force field file. Angle-Angle terms are defined for 3 atoms that are all bonded to a single central atom. The two angles are inaa2-current-inaa3 and inaa2-current-inaa4.

This section is repeated based on the Number of Angle-Angle Terms

'Angle-Angle Type Number' (a23) [integer]

- The number coresponding to the angle-angle type. These numbers must run from 1 to the Number of Angle-Angle Terms.

'Angle-Angle Style' (a17) [integer]

- The number coresponding to the angle-angle style. Below is a list of Angle-Angle Styles, their potential form, and the aacoeffs. The angle-angle potentials have a variety of forms (as shown below). Any energy terms need to use the standard code units of Kelvins. All angles are computed in radians within Towhee, so any additive terms with the angle-angles should also be listed in radians.
  - 1: Compass Angle-Angle with autodetect
    $U_{angle-angle}$ = aacoeff(0) * [angle(inaa0-current-inaa1) - bencoeff(inaa0-current-inaa1)] * [angle(inaa0-current-inaa2) - bencoeff(inaa0-current-inaa2)]
  - 2: Compass Angle-Angle with explicit parameters
    $U_{angle-angle}$ = aacoeff(0) * [angle(inaa0-current-inaa1) - aacoeff(1)] * [angle(inaa0-current-inaa2) - aacoeff(2)]

'Angle-Angle Coefficients' (a24) [double precision]

- The aacoeff variables, listed one per line, starting with the lowest index variable and working upwards.

'Force Field Name' (a16) [character string a10]
- The name of the Force Field

'Number of Atoms with Same Parameters' (a38) [integer]
- The number of quartets of atoms in this force field which use the same improper parameters

'Atom Names' (a10) [character string quartets a10,1x,a10,1x,a10,1x,a10]
- All of the quartets of Atom Names for this Angle-Angle Type. These are listed with one quartet of atom names per line. The Compass naming convention is followed to add ease of transfer from their publications. The order for atom output in this convention is inaa1, current, inaa0, inaa2.

End of the section repeated based on the Number of Angle-Angle Terms

---

'Number of One-Five Types' (a23) [integer]
- The number of different special One-Five interaction types the force field file.

This section is repeated once for each of the Number of One-Five Types

'One-Five Type Number' (a20) [integer]
- The number coresponding to the special one-five interaction type. These numbers must run from 1 to the Number of One-Five Types.

'One-Five Style' (a14) [integer]
- The number coresponding to the special one-five interaction style. Below is a list of One-Five Styles, their potential form, and the ofcoeffs. Currently there is only one kind of one-five potential. Any energy terms need to use the standard code units of Kelvins.
  - 1: Lennard-Jones 12-6 interactions with special parameters

    $U_{nonbond}$ = 4 * ofcoeff(2) * [(ofcoeff(1)/r)^{12} - (ofcoeff(1)/r)^{6} - Shift]

    Where the parameters are as follows.
    - ofcoeff(1): sigma in units of Angstroms.
    - ofcoeff(2): epsilon in units of Kelvin.
    - Shift: computed automatically by the code if we are using a shifted potential.

'One-Five Coefficients' (a22) [double precision]
- The ofcoeff variables, listed one per line, starting with the lowest index variable and working upwards.

'Force Field Name' (a16) [character string a10]
- The name of the Force Field

'Atom Names' (a10) [character string quintet a10,1x,a10,1x,a10,1x,a10,1x,a10]
- The five atom names that, when bonded successively, create a special one-five interaction. These names are compared with the **nbnames** to find a valid match. If no matches are found then it is assumed that no special one-five interaction are needed for these atoms and the normal method of computing nonbonded pair interactions is used instead.

End of the section repeated based on the Number of One-Five Types

---

'Number of Bond Increments' (a25) [integer]
- The number of different Bond Increment terms in the force field file.

This section is repeated once for each of the Number of Bond Increments

'Bond Increment Type Number' (a26) [integer]
- The number coresponding to the bond increment type. These numbers must run from 1 to the Number of Bond Increments.

'Bond Increment Value' (a20) [double precision]
- The value of the bond increment. This value is added to the charge of the first atom listed in the **Atom Names** and subtracted from the charge of the second atom listed in the **Atom Names**.

'Bond Increment Order' (a20) [character string a10]
- The bond increment order. This is used to distinguish between different types of bonds that can be made between the same pairs of atoms in the same manner as the 'Vibration Order'. Commonly used

options are listed in the inpstyle 6 manual.

'Force Field Name' (a16) [character string a10]

- The name of the Force Field

'Atom Names' (a10) [character string pair a10,1x,a10]

- The pair of atoms that share this bond increment. The bond increment is added to the base charge of the first atom listed and subtracted from the base charge of the second atom listed when these atoms are connected by a bond.

End of the section repeated based on the Number of Bond Increments

---

Return to the main towhee web page

Send comments to: Marcus G. Martin

Last updated: January 17, 2007

MCCCS Towhee (towhee_parallel)

Overview

This section described the format that is used with the parallel version of the Towhee code (compiled with --enable-mpi flag). If invoked with the "jobfarm" parallel style (with "-p jobfarm" flag, or no "-p" flag at all), towhee reads from an additional file (towhee_parallel) which describes the parallelization strategy to employ for the simulation. Only the current version of this manual is maintained. This was last updated for version 5.0.1.

Several other parallel styles exist and they are documented in the code. To learn more about these options compile the parallel version of towhee and invoke it with the following command.

towhee -h

Variable explanations for towhee_parallel

number of jobs (integer)

- The number of simulations to run. This is the **m** variable in the above equation

stdout filename (formatted character string of length 30 or less)

- The filename for the output that would normally go to stdout. This output is placed in each of the working directories.

working directories (formatted character string of length 100 or less)

- The list of working directories (one per line) where all of the towhee input files are located for each job that you wish to perform. All of the output will be placed in these directories in a manner identical to if you were running a serial job from that directory. Please note that you should specify the full path name for the ff_filename variable (associated with the force field files) in towhee_input.

Return to the main towhee web page

Send comments to: Marcus G. Martin

Last updated: November 24, 2006

MCCCS Towhee (Standard Output)

Overview

This section provides some information about the quantities computed and displayed to the standard output. Much of the output to the standard output is self-explanatory, so here we focus on the quantities computed and output as averages or as block averages. This section is valid for the most current version of the code and was last updated for version 5.0.1.

Chemical Potential

There are several different types of chemical potential output by the Towhee program. Prior to the 4.5.2 release only the Gibbs total chemical potential has the same meaning as it does currently. The various chemical potentials reported by the code are listed and defined here. Consider the following equation as a starting point.

$\mu_{total}(i) = \mu_{residual}(i) + \mu_{density}(i) + \mu_{isolation}(i)$

$\mu_{total}(i)$ is the total chemical potential of molecule **i** and is the appropriate value to consider when looking for total free energies of a system. This total chemical potential is then commonly broken down into several subcomponents either for computational convenience, or because the subcomponents relate to certain thermodynamic quantities. Towhee defines, and calculates, these subcomponents as follows.

- The total Gibbs chemical potential is computed using the insertion Rosenbluth weight (containing all interactions - intermolecular and intramolecular) and is computed each time a multi-box swap move is attempted, and also **chempotperstep** times at the end of every Monte Carlo cycle.

  $\mu_{total}(i) = - k_B T \ln[< W * V / ([N(i)+1] * \Lambda^3(i)) >]$

  where $\mathbf{k_B}$ is Boltzmann's constant, **T** is the temperature, **V** is the volume of the simulation box, **N(i)** is the number of molecules of type **i** in the simulation box, and **Λ(i)** is the thermal de Broglie wavelength for molecule type **i**. The < > brackets indicate an ensemble average of the quantity inside the brackets.
- The ideal density dependent portion of the chemical potential is

  $\mu_{density}(i) = - k_B T \ln[< V / N(i) > / \Lambda^3(i))]$
- The isolated molecule portion of the chemical potential is the non-density dependent chemical potential of an isolated molecule in an ideal gas (which means it does not interact with any other molecule, but does interact with itself). This quantity is not normally computed by Towhee. However, this is computed in the special case where the total number of molecules (**nmolectyp**) of one of the species is set to zero. In that case, the isolated molecule chemical potential is computed **chempotperstep** times per cycle. It uses the full Rosenbluth weight (**W**) computed for growing the molecule in an extremely large box that has the Ewald sum and the van der Waals tail corrections disabled. This uses the standard formula for computing the chemical potential via configurational-bias.

  $\mu_{isolation}(i) = -k_B T \ln[< W_{isolation} >]$.
- The NVT insertion chemical potential is computed in a very similar manner as the Gibbs total chemical potential. The only difference is the lack of any number density terms in the average. This chemical potential calculation is only formally correct for a single box canonical ensemble. However, experience has shown that this chemical potential is often better when the average number of molecules of the type of interest drops below 1. It turns out that systematic errors in the number density term from using the Gibbs total chemical potential (essentially the use of N+1) are significant in this case. This is discussed further in Martin and Siepmann 1998 (TCA).

  $\mu_{NVT\ Insertion}(i) = - k_B T \ln[< W >]$
- The NpT insertion chemical potential is computed in a very similar manner as the NVT insertion chemical potential. The only difference is the inclusion of a volume term in the average, that is then removed later by dividing out the average volume. This chemical potential calculation is only formally correct for a single box isobaric-isothermal ensemble. However, experience has shown that this chemical potential is often better when the average number of molecules of the type of interest drops below 1. It turns out that systematic errors in the number density term from using the Gibbs total chemical potential (essentially the use of N+1) are significant in this case. This is discussed further in Martin and Siepmann 1998 (TCA).

  $\mu_{NpT\ Insertion}(i) = - k_B T \ln[< V W > / < V >]$
- The residual chemical potential is not computed directly in Towhee. Instead it is inferred by computing the full insertion chemical potential and then substracting the isolated molecule chemical potential. This chemical potential is only computed in the special case when we are interested in Henry's Law coefficients and **nmolectyp** is set to 0 for at least one of the components. The residual chemical potential is computed in two slightly different ways depening on the ensemble. If we are using the canonical ensemble then the residual chemical potential is

  $\mu_{residual}(i) = \mu_{NVT\ Insertion}(i) - \mu_{Isolation}(i)$

  and if we are using the isobaric-isothermal ensemble (the typical ensemble for computing Henry's law) then the residual chemical potential is

  $\mu_{residual}(i) = \mu_{NpT\ Insertion}(i) - \mu_{Isolation}(i)$.

Pressure

Pressure is computed using several different methods in Towhee and this section contains a brief description of these methods.

- **Ideal Pressure**

  This method of computing the pressure takes the total number density of molecules in a box and then converts that into a pressure using the ideal gas law (pV = nRT). This works fairly well for vapors, but is very poor for condensed phases. This is computed mostly as a reference point as quantities such as the compressibility are expressed as a ratio of the true pressure to the ideal pressure. However, it is useful for systems such as polyatomic molecules with discontinuous potentials where none of the other methods are correct either. This is computed at the end of the simulation by transforming the average total number density into a pressure using the ideal gas law.
- **Radial Pressure**

  This method of computing the pressure uses an estimate of the radial distribution function at the locations just below and just above a discontinuity in the intermolecular potential and combines that with the molecule density to compute what I call the Radial Pressure. It is based upon the standard method of computing the pressure for discontinous systems, as described in section 6.5.3 of Allen and Tildesley with some modifications so it works on the fly instead of as a post-processing step. Only a small portion of the radial distribution function is used to compute the pressure for discontinous potentials, and those are immediately adjacent to the discontinuity.

$$p_{\text{radial}} = \left(kT \sum_{i=1}^{n_{\text{types}}^{\text{molecule}}} \rho_i \right) - \left(\frac{2}{3}\pi \sum_{j=1}^{n_{\text{types}}^{\text{atom}}} \sum_{k=1}^{n_{\text{types}}^{\text{atom}}} \sum_{x=1}^{n_{\text{discontinuities}}} r_x^3 \left[g_{jk}(r_x^-) - g_{jk}(r_x^+) \right] \right) \qquad Eq.1$$

This equation is a generalization of equation 7b in Smith *et al.* 1977. The Towhee implementation computes the intermolecular radial distribution function only for bins adjacent to the discontinuities, where the bin width is set by **radial_pressure_delta** in towhee_input. This method is exact in the limit where **radial_pressure_delta** approaches zero, but in practice that limit results in very poor statistics as it gets progressively less likely to observe atoms in a bin of a certain distance when the width of that bin goes to zero. It also works exactly for polyatomic molecules that are hard spheres of zero radius (what I call an ideal chain). I am unsure whether this implementation is correct for polyatomic molecules that are not just hard points and the developers welcome feedback from anyone who has a better understanding of the proper formulation of the pressure for hard sphere or square well molecules.

- **Thermodynamic Pressure**
  This method of computing the pressure uses the average change in energy per trial change in volume as an estimate of the thermodynamic quantity dU/dV and then combines that with the number density in order to get a pressure. It is implemented as described in Hummer *et al.* 1998 with the exception that any attempted move that would result in an infinite energy (occurs frequently for discontinous potentials like Hard Spheres or Square Wells, and also infrequently for continous potentials with sufficiently large volume attempts depending upon the value of **rmin**). The dU/dV term is averaged after every attempted volume change move (without regard to whether the move was accepted) and then combined with the average number density to compute the thermodynamic pressure.
- **Virial Pressure**
  The molecular virial, described in detail in chapter 2 of Allen and Tildesley is used to compute the pressure from the intermolecular pair forces in the system. This is the default method of computing the pressure in most Monte Carlo simulations and works well so long as the intermolecular pair forces are continuous and relatively easy to derive. It does not work for step potentials (such as Hard Sphere or Square Well) and is also not implemented for potentials that have an intermolecular angle terms (such as Stillinger-Weber). This method computes the pressure for individual snapshots of the system with a frequency controlled by the **pressurefreq** variable.

Average Accumulation

The following quantities are added into the averages after every attemped Monte Carlo move.
- Total energy
- Individual energy components
- Total energy squared
- Specific density
- Volume
- Number density of each component and also the total number density
- Mol fraction
- Number of each type of molecule in each box
- Heat of vaporization (pV/n=RT)

The following quantities are added into the averages after every full Monte Carlo cycle.
- Radius of Gyration

The following quantities are added into the averages every time the virial pressure is computed.
- Virial Pressure. The pressure in Towhee is computed using the molecular virial.
- Stress tensor
- Enthalpy. Computed in Towhee using the thermodynamic relation H = U + pV.
- Enthalpy squared
- pV (pressure times volume)
- Heat of vaporization (Direct)
- Heat of vaporization (vapor p)

The following quantities are added into the averages every time a volume move is attempted and it does not result in an infinite energy change (hard overlap)
- dU/dV: the change in potential energy for a trial change in volume. This is used to compute the thermodynamic pressure.

The following quantities are computed every time the appropriate chemical potential is measured
- Isolation chemical potential.
- NVT insertion chemical potential.
- NpT insertion chemical potential.
- Gibbs total chemical potential

The following quantities are computed using the values of other quantities
- Ideal density chemical potential (**u (Density)**). Computed using the average volume and the average number of molecules.
- Henry's law coefficients. This is computed from the residual chemical potential and the number density.

  Henry(i) = k_B T < $\text{Sum}_{j=1}^{n}$\{ number density(j) \} > * Exp[< μ_{residual}(i) > / k_B T]
- Ideal Pressure. Computed from the average total number density.
- Thermodynamic Pressure. Computed using the Ideal Pressure and dU/dV following the work of Hummer *et al.* 1998.

Return to the main towhee web page

Send comments to: Marcus G. Martin

Last updated: December 14, 2006

MCCCS Towhee Utility

MCCCS Towhee (analyse_histogram)

Overview

This section explains the analyse_histogram utility program. This program is designed for use in conjunction with the towhee_histogram files and is used to analyze the histogram data generated by the grand canonical ensemble.

Compiling

analyse_histogram.F is a stand alone program so on a unix machine you would compile the code using a statement similar to

```
gfortran analyse_histogram.F -o analyse_histogram.x
```

This will create the analyse_histogram.x executable which can then be run from the command line. In order to successfully run analyse_histogram, two sources of input are required. The first is an input file named 'input_analyse_histogram', which contains parameters that define what analysis you would like to perform. These parameters will be defined in the following section. The second input is the histogram files which are generated by performing simulations of the grand canonical ensemble in towhee (towhee_histogram).

Features

Run the analyse_histogram.x utility routine by calling the routine from a directory that contains towhee_histogram output files from a MCCCS Towhee simulation run and a copy of the input_analyse_histogram which is setup so that your desired calculations will be performed. Below is a list of the calculations that analyse_histogram can perform and the representative file names that will contain the results.

- histogram reweighting (lweight is .true.): this computes the weights for the histograms using the method of Ferrenberg and Swendsen 1989 The weights are output into a file called towhee_weights, which can be used as input in the lold option. Also, converge.dat, which contains the number of iterations and the deviation, is outputted during these calculations so that you can see how well the algorithm is converging.
- phase coexistence calculations (lphase is .true.): this computes the phase coexistence for binary mixtures using Newton's method. The phase coexistence data is output in 'phase_coex.dat.' The distribution of the number of each component *n* at each temperature *t* is output in a file called 'phasecomp*n*temp*t*.dat.'
- pvt calculations (lpvt is .true.): calculates ln *Z* vs. *n* (number of molecules) data so that the constant in 'pv/T = ln xi + constant' may be found. The ln*Z* and *n* data is output in a file called 'pvt.dat'. Also, the distribution of each component *n* at each temperature *t* is output in a file called 'pvtcomp*n*temp*t*.dat.'

Parameter explanations for input_analyse_histogram

ncomp (integer)
- The number of components in the system (Note: only the weight calculation feature of this code has been tested for systems with more than 3 components)

volume (double precision)
- The volume of the box in nm^3. (Note: these are the units it is output in the standard Towhee output at the end of a simulation.)

lweight (logical)
- .true. :weights for the histograms are computed and the following variable is read from this file.

  ndump (integer)
  - The frequency that the iteration and weights are outputted during the

iteration process
  - .false. if weights have alread been calculated for the histograms and do not want to recompute them

lphase (logical)
  - .true. : the phase coexistence is calculated and the following variables are read from a file named 'file_phase'

    chempot1 (double precision)
      - An initial guess for the chemical potential for component 1

    Number Entries (integer)
      - The number of temperatures at which phase coexistence is to be calculated.

    The following section is repeated once for each value of **Number Entries**. [Note: If **ncomp** = 1, then only entries for **Temperature** , **midpt** and **slope** are required.]

    Temperature (double precision)
      - The temperature in Kelvin.

    midpt (double precision)
      - A variable used to split the integratation used for the Newton's Method.

    slope (double precision)
      - The slope is used to split the peaks of the coexistence behavior correctly. An initial guess of 0.0 is okay, and if the phase data looks okay then it should be fine.

    If ncomp = 1 , then this is the end of the section repeated once for each value of **Number of Entries** . *If ncomp = 2* , then each of the following values will also need to be included.

    chempot2min (double precision)
      - The minimum that the chemical potential of component 2 would be.

    chempot2max (double precision)
      - The maximum that the chemical potential of component 2 would be.

    chempot2incr (double precision)
      - The increment at which to investigate the range of chempot2min to chempot2max

    If ncomp = 2 , this is the end of the section repeated once for each value of **Number Entries**

  - .false. : phase coexistence is not calculated

lpvt (logical)
  - .true. :pvt calculations are performed and the following variables are read from a file named 'file_pvt'.

    Number of entries
      - The number of entries at which pvt calculations are to be conducted,

    ---

    The following section is repeated once for each value of **Number of entries**

    Temperature (double precision)
      - The temperature in Kelvin.

    Chemical potential (double precision)
      - The chemical potential of each component of the system is given one after another.

    End of the section that is repeated once for each value of **Number of entries**

  - .false. if pvt calculations are not to be performed

lold (logical)
  - .true. if you are including histograms which have already had their weights determined once before. In that case the following variables must be listed in this file

    noldfiles (integer)

        - The number of histograms that are included for which the weights have already been included.

      file_old (character)

        - The name of the data file of the old runs weights and other needed information. This is the same as the towhee_weights file which is ouputted during the weight calculation. So can just copy this file into a new file and use it. (**Note:** You must list the filename of the histogram for the low density reference state first.)

  - .false. if you have no histograms which already have had their weights determined

lnew (logical)

  - .true. if you are including histograms which still need to have their weights determined. In that case the following variables must be listed in this file.

      nfiles (integer)

        - The number of histograms that still need their weights to be determined.

The following section is repeated once for each value of **nfiles**

filename (character)

- The name of the file containing the histogram data (towhee_histogram) which is output by Towhee. (**Note:** If you do not have old histograms, you must list the filename of the histogram for the low density reference state first.)

number of entries (integer)

- The number of entries in the histogram data file.

End of the section that is repeated once for each value of **nfiles**

  - .false. if you are not including any histograms which still need to have their weights determined

Return to the Utility Summary page

Send comments to: Marcus G. Martin
Last updated: November 07, 2007

MCCCS Towhee (analyse_movie)

Overview

This section explains the analyse_movie utility program. This program is designed for use in conjunction with the Towhee MCCCS file and is used to compute quantities from the towhee_movie file. This manual is accurate only for the current version (2) and was last updated for code version 4.8.1.

Compiling

analyse_movie.F is a stand alone program so on a unix machine you would compile the code using a statement similar to

```
gfortran analyse_movie.F -o analyse_movie.x
```

This will create the analyse_movie.x executable which can then be run from the command line. Note that there are a few parameters that are set at the top of the file that you may have to modify. These values have meanings quite similar to those which appear in preproc.h

Features

Run the analyse_movie.x utility routine by calling the routine from a directory that contains the towhee_movie output file from a MCCCS Towhee simulation run. You will then be asked a series of questions about whether you wish to compute certain quantities. Answer the questions with T if you wish to compute that quantity, or F if you do not wish to compute that quantity. Below is a list of the quantities that analyse_movie can compute. Note that the output files generated by analyse_movie all have a similar structure where the characters at the beginning are a shorthand name for the file, then there is an _b### where ### is the simulation box number, and then many of the files also have a _t### where this time ### is the molecule type number, followed by a sequence of _u### numbers which are the unit numbers of the atoms in the various interactions.. For example, the torsion distribution in box 1 for molecule type 1 between atoms 2, 3, 4, and 8 would be tor_b001_t001_u002_u003_u004_u008.

- radial distribution functions: this computes the intermolecular radial distribution functions (rdf) between all types of atoms.
- atom distribution profiles: this computes the probability of finding each atom of each type of molecule in the x,y, and z dimensions of the simulation boxes. The simulation box size is normallized to 1 so that this quantity would still make sense even if the box size is varying throughout the simulation. The output files start with xpr, ypr, and zpr for the distributions in the x, y, and z dimensions.
- vibration distribution profiles: computes the vibration distribution for each molecule in each box. Output files start with vib
- bending angle distribution profiles: computes the bending angle distribution for each molecule in each box. Output files start with ben.
- dihedral angle distribution profiles: computes the dihedral angle distribution for each molecule in each box. Output files start with tor.

Return to the Utility Summary page

Send comments to: Marcus G. Martin
Last updated: November 07, 2007

MCCCS Towhee (fitcoex)

Overview

This section explains the fitcoex utility program. This program is designed for use in conjunction with the towhee_vlcc output files in order to compute the critical temperature and critical density from a sequence of single-component vapour-liquid simulations.

Compiling

fitcoex is compiled in a similar manner as **towhee** or **forcefield**.

```
cd /towheebase/Source
make fitcoex
```

This will create the fitcoex executable which can then be run from the command line.

Features

In order to successfully run fitcoex, two sources of input are required. The first is an input file (*.sim), which contains some comment lines, a value for the critical exponent "beta", the towhee_vlcc data from at least 3 different temperatures, and a special end of file zero line. The second file (*.exp) is optional and is for transforming experimental data into a similar format as the simulation data for comparison purposes. An example with the input files and a for plotting the output is provided in the Examples/VLCC_Fit directory.

File Format for the towhee_vlcc compilation file (*.sim)

Line 1 comment line (typically has a comment about how the data was generated)
Line 2 comment line (typically says 'beta' to remind the user that the beta value is on the next line
Line 3 beta (double precision). I typically use a value of 0.325 for this variable
Line 4 to 4+n towhee_vlcc data lines. You must have at least 3 of these lines at different temperatures in order for the code to compute the critical properties.
Line 5+n zero end line. The code only knows it has hit the end of the file when it gets a line that has a zero value for the variables. 5 double precision zeros go on this line.

File Format for the experimental data file (*.exp)

There are currently two experimental data file formats implemented into this routine. These files are translated into units of g/ml and output to several files by fitcoex.

- Smith and Srivastava: This format is similar to the one used in the reference book Smith and Srivastava 1986.
  Line 1 comment line
  Line 2 molecular weight of the molecule (g/mol)
  Line 3 comment line
  Line 4 comment line
  Line 5 to 5+n experimental temperature (K), liquid phase molar volume (ml/mol), and vapor phase molar volume (ml/mol).
- Hensel and Warren: This format is similar to the one used in the reference book Hensel and Warren 1999.
  Line 1 comment line
  Line 2 molecular weight of the molecules (g/mol)
  Line 3-4 comment lines
  Line 5 to 5+n experimental temperature (C), saturation pressure (bar), liquid density (g/ml), vapor density (g/ml)

Run the code by typing

```
/towheebase/Source/fitcoex
```

Return to the Utility Summary page

Send comments to: Marcus G. Martin

Last updated: November 24, 2006

MCCCS Towhee (parse_vlcc_plots script)

Overview

This section explains the parse_vlcc_plots utility script. This BASH script is designed for use in conjunction with the standard towhee output file and parsest that file with gawk and grep in order to create several plottable files of the variables that are output during the run and the block averages.

Compiling and Running

parse_vlcc_plots.script is a BASH script and does not need compiling. You just need a BASH shell (the default option on most Linux installations). Please note that this script only works with output files generated from Towhee 3.17.5 or subsequent versions.

Features

If your standard output file is named "towhee_output" then you would parse that file using the following command.

```
./parse_vlcc_plots.script towhee_output
```

This will create a directory named "Plots" which will contain the following files with data extracted from your standard output file.

Files with data extracted from the block averages

- **avgchempot_box1**: contains the chemical potential (In simulation box 1) from the individual blocks used to compute the block averages.
- **avgchempot_box2**: contains the chemical potential (In simulation box 2) from the individual blocks used to compute the block averages.
- **avgden_box1**: contains the specific density (In simulation box 1) from the individual blocks used to compute the block averages.
- **avgden_box2**: contains the specific density (In simulation box 2) from the individual blocks used to compute the block averages.
- **avgeng_box1**: contains the total potential energy (In simulation box 1) from the individual blocks used to compute the block averages.
- **avgeng_box2**: contains the total potential energy (In simulation box 2) from the individual blocks used to compute the block averages.
- **avgnumden_box1**: contains the number density (In simulation box 1) from the individual blocks used to compute the block averages.
- **avgnumden_box2**: contains the number density (In simulation box 2) from the individual blocks used to compute the block averages.
- **avgpres_box1**: contains the pressure (In simulation box 1) from the individual blocks used to compute the block averages.
- **avgpres_box2**: contains the pressure (In simulation box 2) from the individual blocks used to compute the block averages.

Files with data extracted from the snapshots

- **energy_box1**: contains the total energy (In simulation box 1) from the system snapshots.
- **energy_box2**: contains the total energy (In simulation box 2) from the system snapshots.
- **molecules_box1**: contains the number of molecules (In simulation box 1) from the system snapshots.
- **molecules_box2**: contains the number of molecules (In simulation box 2) from the system snapshots.
- **pressure_box1**: contains the pressure (In simulation box 1) from the system snapshots.
- **pressure_box2**: contains the pressure (In simulation box 2) from the system snapshots.
- **volume_box1**: contains the volume (In simulation box 1) from the system snapshots.
- **volume_box2**: contains the volume (In simulation box 2) from the system snapshots.

Return to the Utility Summary page

Send comments to: Marcus G. Martin
Last updated: September 15, 2004

MCCCS Towhee (charmm2pdb)

Overview

This section explains the charmm2pdb utility program. This program is designed to translate the polypeptide pdb files generated by Charmm into the standard pdb format to allow further processing via the pdb2towhee routine.

Compiling

charmm2pdb.F is a stand alone program so on a unix machine you would compile the code using a statement similar to

```
gfortran charmm2pdb.F -o charmm2pdb.x
```

This will create the charmm2pdb.x executable which can then be run from the command line.

Features

When you run this program from the command line it will prompt you to enter the name of the charmm pdb file that you wish to translate. It will then ask you the number of atoms in this file. Please note that you must clean up the charmm pdb file so that the first line of the file starts right in with the list of atoms. This utility will then output a file named **standard.pdb** which will have the peptide atoms listed in the standard pdb order instead of the default charmm order.

Return to the Utility Summary page

Send comments to: Marcus G. Martin
Last updated: November 07, 2007

MCCCS Towhee (faux2towhee)

Overview

This section explains the faux2towhee utility program. This program is designed to translate some zeolite coordinate files that were kindly provided to me by David Faux into a format suitable for Towhee. This utility is unlikely to be of use to anyone else, but may serve as an example for other one-time file translations.

Compiling

faux2towhee.F is a stand alone program so on a unix machine you would compile the code using a statement similar to

```
gfortran faux2towhee.F -o faux2towhee.x
```

This will create the faux2towhee.x executable which can then be run from the command line.

Features

This is a very crude routine and requires the user to edit the Fortran code in order to set variables like the file names and Towhee inpstyle. It requires two files named faux_4a.atoms and faux_4a.ang and it outputs a towhee_altinp and towhee_coords file that were used to setup the zeolite portion of the Catlow_Zeolite_4a example.

Return to the Utility Summary page

Send comments to: Marcus G. Martin
Last updated: November 07, 2007

MCCCS Towhee (forcefield)

Overview

This section explains the forcefield utility program. This program is designed to translate the the native units of many different forcefields into the Towhee units and then output a towhee_ff file with the current format.

Compiling

Unlike most of the other utility programs, forcefield is generated in a similar way as the main towhee program because it shares several subroutines with towhee. Some of the subroutines in this utility program are very large and cause many common optimizers to overload memory. Therefore I strongly suggest that you turn off the optimizers when compiling this utility. The following direction work and assume you are starting in the base directory of the distribution.

```
./configure FFLAGS="
cd Source
make clean
make forcefield
```

This will create the forcefield executable which can then be run from the command line.

Features

When you run this program from the command line it will prompt you to choose from a list of force field names to generate. Most of these commands simply translate the appropriate built in subroutine of data to output a force field file that should be identical to the one provided with the distribution. Here I list some of the other options that do something other than just generate the specified force field file.

- **All**: this command generates all of the force field files that are simple translations of the subroutines that are included in the main Towhee distribution. This command is useful for regenerating all of those forcefield files after making a change to the force field format.
- **CharmmFile**: translates certain kinds of charmm force field files into a Towhee force field file.
- **PMF**: this command translates a sequence of potential of mean force (PMF) files into a tabulated potential and outputs a towhee_ff_PMF file. For more information about generating PMF file please see the rdf2pmfpair utility.
- **Setfl**: this command translates an Embedded Atom Method potential in the Setfl format into the Towhee format.

Return to the Utility Summary page

Send comments to: Marcus G. Martin
Last updated: October 11, 2004

MCCCS Towhee (pdb2towhee)

Overview

This section explains the pdb2towhee utility program. This program is designed to translate the standard pdb files into appropriate Towhee files for starting a simulation. In all cases it will output a **towhee_coords** file full of the system coordinates. In certain cases it will output a **towhee_altinp** file with partial information required to set up a polypeptide molecule template.

Compiling

pdb2towhee.F is a stand alone program so on a unix machine you would compile the code using a statement similar to

```
gfortran pdb2towhee.F -o pdb2towhee.x
```

This will create the pdb2towhee.x executable which can then be run from the command line.

Features

When you run this program from the command line it will ask you a number of questions about the file that you wish to translate. Those questions and a discussion of the available options are listed here.

Please enter the type of input file (integer)

- **1 - protein pdb file**
  This requires the input pdb file to be in the standard format, to have had all of the header information removed (the first line should be the first ATOM line), and to contain a polypeptide that you wish to input into Towhee using the polypeptide builder. This option will output a towhee_coords file with the system coordinates and a towhee_altinp file with the listing of amino acids in the Towhee code. You will need to sort through the towhee_altinp afterwards to sort out the protonation states of histadine, and to assign any bonding partners for disulfide bridges.
- **2 - generic pdb file**
  This requires the input pdb file to be in the standard format, and to have had all of the header information removed (the first line should be the first ATOM line). This option will output a towhee_coords file with the system coordinates.
- **3 - generic Charmm output file**
  This requires the input pdb file to be in the Charmm format and to have had all of the header information removed (the first line should be the that contains coordinates). This option will output a towhee_coords file with the system coordinates.
- **4 - Charmm output file (DNA)**
  This requires the input pdb file to be in the Charmm format, to be a DNA strand, and to have had all of the header information removed (the first line should be the that contains coordinates. This option will output a towhee_coords file with the system coordinates and a towhee_altinp file with the listing of DNA bases in the Towhee code.
- **5 - Charmm output file (RNA)**
  This requires the input pdb file to be in the Charmm format, to be a RNA strand, and to have had all of the header information removed (the first line should be the that contains coordinates. This option will output a towhee_coords file with the system coordinates and a towhee_altinp file with the listing of RNA bases in the Towhee code.

Please enter the pdb file name (character string)

The name of the file that you wish to translate

Please enter the number of atoms (integer)

The number of atoms that you wish to translate from the old file into the Towhee files.

Return to the Utility Summary page

Send comments to: Marcus G. Martin
Last updated: November 07, 2007

MCCCS Towhee (rdf2pmfpair)

Overview

This section explains the rdf2pmfpair utility program. This program converts a radial distribution function into a potential of mean force. The potential of mean force file is then suitable for conversion into a tabulated pair potential using the forcefield program.

Compiling

rdf2pmfpair.F is a stand alone program so on a unix machine you would compile the code using a statement similar to

```
gfortran rdf2pmfpair.F -o rdf2pmfpair.x
```

This will create the rdf2pmfpair.x executable which can then be run from the command line.

Features

When you run this program from the command line it will prompt you to enter the name of the radial distribution function file that you wish to translate. It will then ask you the number of lines of data in this file, the temperature, and the nonbonded cutoff. This utility will then output a file named **towhee_pmf** that translates the radial distribution function into a potential of mean force using

$$PMF(r) = -k_B\ T \ln[\ g(r)\ ]$$

where PMF(r) is the potential of mean force and g(r) is the radial distribution function.

Return to the Utility Summary page

Send comments to: Marcus G. Martin
Last updated: November 07, 2007

MCCCS Towhee (unitcell)

Overview

This section explains the unitcell Towhee utility program. This program is used to take a towhee_coords file and duplicate it any number of times in the x, y, and z dimensions in order to perform a larger simulation.

Compiling

unitcell.F is a stand alone program so on a unix machine you would compile the code using a statement similar to

```
gfortran unitcell.F -o unitcell.x
```

This will create the unitcell.x executable which can then be run from the command line.

Features

When you run this program from the command line it will ask you a number of questions about the towhee_coords file that you wish to duplicate. Those questions and a discussion of the available options are listed here. The input file is **towhee_coords** and the output file is **towhee_newcoords**.

Enter the original unit cell x-dimension (double precision)
: The length of the box in the x-dimension in the original unit cell.

Enter the original unit cell y-dimension (double precision)
: The length of the box in the y-dimension in the original unit cell.

Enter the original unit cell z-dimension (double precision)
: The length of the box in the z-dimension in the original unit cell.

Enter the number of atoms in the original unit cell (integer)
: The number of atoms in the original **towhee_coords** file.

Enter the number of new duplicates in the x-dimension (integer)
: The number of times to reproduce this unit cell in the x-dimension.

Enter the number of new duplicates in the y-dimension (integer)
: The number of times to reproduce this unit cell in the y-dimension.

Enter the number of new duplicates in the z-dimension (integer)
: The number of times to reproduce this unit cell in the z-dimension.

Return to the Utility Summary page

Send comments to: Marcus G. Martin
Last updated: November 07, 2007

MCCCS Towhee (xmd2towhee)

Overview

This section explains the xmd2towhee Towhee utility program. This program converts EAM force field files that are in the XMD format into a format suitable for use in Towhee.

Compiling

xmd2towhee.F is a stand alone program so on a unix machine you would compile the code using a statement similar to

```
gfortran xmd2towhee.F -o xmd2towhee.x
```

This will create the xmd2towhee.x executable which can then be run from the command line.

Features

When you run this program from the command line it will ask you for the name of an XMD format file to read in and it will output Towhee force field information to the file **towhee_eam**. In order for this translation routine to function properly you must reorder the XMD files slightly. The data in the XMD file must first be the pair potential information, then the embedding, and finally the densities.

Please enter a filename

The name of the XMD file you wish to translate into Towhee format.

Return to the Utility Summary page

Send comments to: Marcus G. Martin
Last updated: November 07, 2007

MCCCS Towhee (xyz2towhee)

Overview

This section explains the xyz2towhee Towhee utility program. This program converts simple xyz format files into a series of files that can be used to construct a **towhee_coords** file for Towhee.

Compiling

xyz2towhee.F is a stand alone program so on a unix machine you would compile the code using a statement similar to

```
gfortran xyz2towhee.F -o xyz2towhee.x
```

This will create the xyz2towhee.x executable which can then be run from the command line.

Features

When you run this program from the command line it will ask you for the name of an XYZ format file to read in and it will output coordinate file to a series of default files labelled something like **fort.41**, **fort.42**, etc. where the final digit of the filename corresponds to the molecule type in the original XYZ file.

Please enter a filename

The name of the XYZ file you wish to translate into Towhee format.

Return to the Utility Summary page

Send comments to: Marcus G. Martin
Last updated: November 07, 2007

MCCCS Towhee (towhee_input classical_potential)

classical_potential (character*30)

The different settings for **classical_potential** require a different set of variables afterwards. For each **classical_potential** a description of the nonbonded potential and the set of subsequent required variables is listed. This documentation is only kept up to date for the most current release version. Last updated for version 5.0.1.

- '12-6 plus 12-10 H-bond': 12-6 Lennard-Jones plus a 12-10 hydrogen bonding term on certain pairs
  This uses the standard Lennard-Jones 12-6 plus an extra 12-10 term between certain pairs of atoms that are identified as hydrogen bond donors and accepters. See Weiner *et al.* 1984 for more information.
  If the two atoms are separated by more than 3 bonds, or are on different molecules then
  $U_{nonbond}$ = 4 * nbcoeff(2) * [(nbcoeff(1)/r)^{12} - (nbcoeff(1)/r)^{6}] + hbondcoeff(1) / r^{12} - hbondcoeff(2) / r^{10}
  else if the two atoms are separated by exactly 3 bonds then
  $U_{nonbond}$ = 4 * nbcoeff(4) * [(nbcoeff(3)/r)^{12} - (nbcoeff(3)/r)^{6}] + hbondcoeff(1) / r^{12} - hbondcoeff(2) / r^{10}
  classical_mixrule (character*30)
  This variable specifies the manner in which the parameters for unlike atoms are determined.
  - 'Lorentz-Berthelot': arithmetic mean of s values [nbcoeff(1) or nbcoeff(3)], geometric mean of e values [nbcoeff(2) or nbcoeff(4)].
  - 'Shukla': complicated expression involving the polarizability of each atom. See equations 19 and 20 of Shukla 1987 for the definition of this mixing rule.

  lshift See the lshift section for information about this variable
  ltailc See the ltailc section for information about this variable
  rmin See the rmin section for information about this variable
  rcut See the rcut section for information about this variable
  rcutin See the rcutin section for information about this variable
- '12-6 plus solvation': 12-6 Lennard-Jones with implicit solvent
  This uses the standard Lennard-Jones 12-6 plus an extra implicit solvation term. See Lazaridis and Karplus 1999 for more information.
  If the two atoms are separated by more than 3 bonds, or are on different molecules then
  $U_{nonbond}$ = 4 * nbcoeff(2) * [(nbcoeff(1)/r)^{12} - (nbcoeff(1)/r)^{6}]
  else if the two atoms are separated by exactly 3 bonds then
  $U_{nonbond}$ = 4 * nbcoeff(4) * [(nbcoeff(3)/r)^{12} - (nbcoeff(3)/r)^{6}]
  classical_mixrule (character*30)
  This variable specifies the manner in which the parameters for unlike atoms are determined.
  - 'Geometric': geometric mean of all nonbonded coefficients.
  - 'Explicit': cross terms explicitly defined in the appropriate towhee_ff file.

  lshift See the lshift section for information about this variable
  ltailc See the ltailc section for information about this variable
  rmin See the rmin section for information about this variable
  rcut See the rcut section for information about this variable
  rcutin See the rcutin section for information about this variable

---

- '12-9-6': 12-9-6 potential
  This variant of the Lennard-Jones family of potentials was implemented for force fields that use a combination of 12-6 and 9-6.
  v(r) = nbcoeff(1) * (1/r)^{12} + nbcoeff(2) * (1/r)^{9} + nbcoeff(3) * (1/r)^{6}
  classical_mixrule (character*30)
  This variable specifies the manner in which the parameters for unlike atoms are determined.
  - 'Geometric': geometric mean of the nonbonded coefficients.
  - 'Explicit': cross terms explicitly defined in the appropriate towhee_ff file.

  lshift See the lshift section for information about this variable
  ltailc See the ltailc section for information about this variable

rmin See the rmin section for information about this variable
rcut See the rcut section for information about this variable
rcutin See the rcutin section for information about this variable

---

- '9-6': 9-6 potential
  If the two atoms are separated 3 or more bonds, or are on different molecules then
  $U_{nonbond}$ = nbcoeff(2) * [2*(nbcoeff(1)/r)^9 - 3*(nbcoeff(1)/r)^6]
  classical_mixrule (character*30)
  This variable specifies the manner in which the parameters for unlike atoms are determined.
    - 'Sixth Power': sixth order combination of s and e of Waldman and Hagler 1993.

  lshift See the lshift section for information about this variable
  ltailc See the ltailc section for information about this variable
  rmin See the rmin section for information about this variable
  rcut See the rcut section for information about this variable
  rcutin See the rcutin section for information about this variable

---

- 'Buffered 14-7': Buffered 14-7 potential
  If the two atoms are separated 3 or more bonds, or are on different molecules then
  $U_{nonbond}(r) = e_{IJ}$ [(1.07 R_{IJ}) / (r + 0.07 R_{IJ})] * [(1.12 $R_{IJ}{}^7$) / (r^7 + 0.12 $R_{IJ}{}^7$)]
  where the e_{IJ} and R_{IJ} parameters are determined for each pair of interactions (I and J) using the mixing rules. The nonbonded coefficients required for this forcefield are as follows.
    - nbcoeff(1): alpha-i in the MMFF94 native units
    - nbcoeff(2): N-i in the MMFF94 native units
    - nbcoeff(3): A-i in the MMFF94 native units
    - nbcoeff(4): G-i in the MMFF94 native units

  classical_mixrule (character*30)
  This variable specifies the manner in which the parameters for unlike atoms are determined.
    - 'MMFF': the mixing rules used to determine e_{IJ} and R_{IJ} from the four nonbonded coefficients as described in the second MMFF94 paper (Halgren 1996 (II)).

  ltailc See the ltailc section for information about this variable. This option is not yet implemented properly as it is surprisingly difficult to integrate the Buffered 14-7 form, but hopefully it will be functioning soon.
  rmin See the rmin section for information about this variable
  rcut See the rcut section for information about this variable
  rcutin See the rcutin section for information about this variable

---

- 'Double Exponential': Double exponential potential
  If the two atoms are separated by 3 or more bonds, or are on different molecules then
  $U_{nonbond}$ = nbcoeff(1) * Exp[- nbcoeff(2) r] - nbcoeff(3) * Exp[- nbcoeff(4) r]
  classical_mixrule (character*30)
  This variable specifies the manner in which the parameters for unlike atoms are determined.
    - 'Explicit': cross terms explicitly specified in the appropriate towhee_ff file.

  lshift See the lshift section for information about this variable
  ltailc See the ltailc section for information about this variable
  rmin See the rmin section for information about this variable
  rcut See the rcut section for information about this variable
  rcutin See the rcutin section for information about this variable

---

- 'Embedded Atom Method': Embedded Atom Method (see Daw and Baskes 1983)
  This is an atomic potential and can only be used with monatomic molecules in Towhee. Historically, the Embedded Atom Method (EAM) uses a lookup table for computing intermolecular interactions and this is an option in Towhee. When tabular data is used the **interpolatestyle** determines the method for interpolating between the specified values. Other functional forms are allowed as described in the Towhee Force Field Documentation. EAM is a short-ranged potential that captures many-body effects by computing

a local density about each atom. This so-called local density is actually a distance dependent function. The sum of the local densities is then fed into the embedding function to compute the embedding energy. Additionally, there is a pair potential term.
interpolatestyle See the interpolatestyle section for information about this variable
rcut See the rcut section for information about this variable

---

- 'EAM pair only': Embedded Atom Method (see Daw and Baskes 1983) using only the pair potential
  This is is the same as the Embedded Atom Method, but only utilizes the pair potential portion and omits the embedding energy. This option exists for academic study of the pair potential portion of the Embedded Atom Method functions. It reads from the same potential files as the 'Embedded Atom Method' potential.
  interpolatestyle See the interpolatestyle section for information about this variable
  rcut See the rcut section for information about this variable

---

- 'Exponential-12-6': Exponential plus 12-6 Lennard-Jones potential
  If the two atoms are separated by more than 3 bonds, or are on different molecules then
  $U_{nonbond}$ = nbcoeff(1)/r^6 + nbcoeff(2)/r^12 + nbcoeff(3)*exp[nbcoeff(4)*r]
  classical_mixrule (character*30)
  This variable specifies the manner in which the parameters for unlike atoms are determined.
    - 'Explicit': cross terms explicitly defined in the appropriate towhee_ff file.

  lshift See the lshift section for information about this variable
  ltailc See the ltailc section for information about this variable
  rmin See the rmin section for information about this variable
  rcut See the rcut section for information about this variable
  rcutin See the rcutin section for information about this variable

---

- 'Exponential-6': Exponential-6 potential
  If the two atoms are separated by more than 3 bonds, or are on different molecules then
  $U_{nonbond}$ = nbcoeff(1)/r^6 + nbcoeff(2) * exp[nbcoeff(3)*r]
  classical_mixrule (character*30)
  This variable specifies the manner in which the parameters for unlike atoms are determined.
    - 'Explicit': cross terms explicitly defined in the appropriate towhee_ff file.

  lshift See the lshift section for information about this variable
  ltailc See the ltailc section for information about this variable
  rcut See the rcut section for information about this variable
  rcutin See the rcutin section for information about this variable

---

- 'Gordon n-6': shifted n-6 potential of Peter Gordon.
  Designed to have the position and depth of the minimum in the potential well to correspond with that of a 12-6 Lennard-Jones potential.
  $U_{nonbond} = 4 * \text{nbcoeff}(2) * c(n) * [\ \{\text{nbcoeff}(1)/(r-a(n))\}^{n} - \{\text{nbcoeff}(1)/(r-a(n))\}^{6}\ ]$
  $a(n) = [\ 2^{1/6} - (n/6)^{1/(n-6)}\ ] * \text{nbcoeff}(1)$
  $r_{minimum} = a(n) + (n/6)^{1/(n-6)} * \text{nbcoeff}(1)$
  $c(n) = -(1/4)*[\ 1 / (\ (\text{nbcoeff}(1)/(r_{minimum} - a(n)))^{n} - (\ \text{nbcoeff}(1) / (r_{minimum} - a(n))\ )^{6}\ )]$
  classical_mixrule (character*30)
  This variable specifies the manner in which the parameters for unlike atoms are determined.
    - 'Lorentz-Berthelot': arithmetic mean of s[nbcoeff(1)], geometric mean of e[nbcoeff(2)]
    - 'Geometric': geometric mean of all nonbonded coefficients.
    - 'Explicit': cross terms explicitly specified in the appropriate towhee_ff file.

  lshift See the lshift section for information about this variable
  ltailc See the ltailc section for information about this variable
  rmin See the rmin section for information about this variable
  rcut See the rcut section for information about this variable
  rcutin See the rcutin section for information about this variable

- 'Hard 2580 Multistep': Hard core multistep potential
  If the two atoms are separated by more than 3 bonds, or are on different molecules, then
  If r < nbcoeff(1)
    $U_{nonbond}$ = Infinity
  else if nbcoeff(1) <= r < 1.2 * nbcoeff(1)
    $U_{nonbond}$ = nbcoeff(2)
  else if 1.2 * nbcoeff(1) <= r < 1.5 * nbcoeff(1)
    $U_{nonbond}$ = nbcoeff(3)
  else if 1.5 * nbcoeff(1) <= r < 1.8 * nbcoeff(1)
    $U_{nonbond}$ = nbcoeff(4)
  else if 1.8 * nbcoeff(1) <= r < 2.0 * nbcoeff(1)
    $U_{nonbond}$ = nbcoeff(5)
  else if 2.0 * nbcoeff(1) <= r
    $U_{nonbond}$ = 0.0
  classical_mixrule (character*30)
    This variable specifies the manner in which the parameters for unlike atoms are determined.
    - 'Lorentz-Berthelot': arithmetic mean of s values, geometric mean of well depths.
  radial_pressure_delta See the radial_pressure_delta section for information about this variable

- 'Hard Sphere': Hard Sphere potential
  If the two atoms are separated by more than 3 bonds, or are on different molecules, then
  $U_{nonbond}$ = Infinity if r <= nbcoeff(1), or 0 otherwise
  classical_mixrule (character*30)
    This variable specifies the manner in which the parameters for unlike atoms are determined.
    - 'Arithmetic': arithmetic mean of s.
  radial_pressure_delta See the radial_pressure_delta section for information about this variable

- 'Lennard-Jones': 12-6 Lennard-Jones potential.
  If the two atoms are separated by more than 3 bonds, or are on different molecules then
  $U_{nonbond}$ = 4 * nbcoeff(2) * [(nbcoeff(1)/r)^{12} - (nbcoeff(1)/r)^{6}]
  else if the two atoms are separated by exactly 3 bonds then
  $U_{nonbond}$ = 4 * nbcoeff(4) * [(nbcoeff(3)/r)^{12} - (nbcoeff(3)/r)^{6}]
  classical_mixrule (character*30)
    This variable specifies the manner in which the parameters for unlike atoms are determined.
    - 'Lorentz-Berthelot': arithmetic mean of s, geometric mean of e
    - 'Geometric': geometric mean of s and e
    - 'Explicit': cross terms explicitly specified in the appropriate towhee_ff file.
    - 'Shukla': complicated expression involving the polarizability of each atom. See equations 19 and 20 of Shukla 1987 for the definition of this mixing rule.
    - 'LB plus manual': uses the Lorentz-Berthelot mixing rules for all atom pairs, except for those that are specified in the following variables.
      mixrule_adjust_total (integer)
        The number of pairs of atoms to set manually. This requires the following variables to be listed **mixrule_adjust_total** times.
        mixrule_adjust_key (character*10)
          - 'element': adjust any pair of nonbonded atom types that match the pair of element names.
          - 'nbname' adjust any pair of nonbonded atom types that match the pair of nonbonded names.
        mixrule_adjust_keynames (character*10 array)
          Character strings that are used to match according to the **mixrule_adjust_key**. These are listed on two lines, one line for each atom in

the pair.

mixrule_adjustments (double precision array)

The values of the nonbonded coefficients to use for this pair of atoms instead of using the Lorentz-Berthelot combining rule. Units should be the same as normal for this type of **classical_potential**. In this case, that is Angstroms for s and Kelvin for e. You need to list four values, one per line: s, e, s (1-4), and e (1-4).

lshift See the lshift section for information about this variable
ltailc See the ltailc section for information about this variable
rmin See the rmin section for information about this variable
rcut See the rcut section for information about this variable
rcutin See the rcutin section for information about this variable

---

- 'Multiwell': multiple square wells and a hard sphere core

  $U_{nonbond}$ = infinity if r <= table_pair(1,1)

  $U_{nonbond}$ = table_pair(2,n) if table_pair(1,n-1) < r <= table_pair(1,n)

  $U_{nonbond}$ = 0 if table_pair(1,npair) < r

  classical_mixrule (character*30)

  This variable specifies the manner in which the parameters for unlike atoms are determined.
  - 'Lorentz-Berthelot': Arithmetic mean for the hard core and square well diameters. Geometric mean for the well depths.
  - 'Explicit': cross terms explicitly defined in the appropriate towhee_ff file.

- 'Repulsive 2580 Multistep': Repulsive core multistep potential

  If the two atoms are separated by more than 3 bonds, or are on different molecules, then

  If r < nbcoeff(1)

  $U_{nonbond}$ = 1.0d10 * (nbcoeff(1)^2 - r^2)

  else if nbcoeff(1) <= r < 1.2 * nbcoeff(1)

  $U_{nonbond}$ = nbcoeff(2)

  else if 1.2 * nbcoeff(1) <= r < 1.5 * nbcoeff(1)

  $U_{nonbond}$ = nbcoeff(3)

  else if 1.5 * nbcoeff(1) <= r < 1.8 * nbcoeff(1)

  $U_{nonbond}$ = nbcoeff(4)

  else if 1.8 * nbcoeff(1) <= r < 2.0 * nbcoeff(1)

  $U_{nonbond}$ = nbcoeff(5)

  else if 2.0 * nbcoeff(1) <= r

  $U_{nonbond}$ = 0.0

  classical_mixrule (character*30)

  This variable specifies the manner in which the parameters for unlike atoms are determined.
  - 'Lorentz-Berthelot': arithmetic mean of s, geometric mean of well depths.

- 'Repulsive Multiwell': multiple square wells with a linear repulsive core

  $U_{nonbond}$ = 1.0d5 * (table_pair(1,1) - r) if r <= table_pair(1,1)

  $U_{nonbond}$ = table_pair(2,n) if table_pair(1,n-1) < r <= table_pair(1,n)

  $U_{nonbond}$ = 0 if table_pair(1,npair) < r

  classical_mixrule (character*30)

  This variable specifies the manner in which the parameters for unlike atoms are determined.
  - 'Lorentz-Berthelot': Arithmetic mean for the hard core and square well diameters. Geometric mean for the well depths.
  - 'Explicit': cross terms explicitly defined in the appropriate towhee_ff file.

- 'Repulsive Sphere': This is used to help setup and equilibrate a hard sphere system where it is sometimes challenging to create an initial conformation with no overlaps. Use this option to equilibrate until the nonbonded potential energy is 0.0, and then switch back to the normal Hard Sphere potential.

If the two atoms are separated by more than 3 bonds, or are on different molecules then
$U_{nonbond}$ = 1d5 + 1d5 * (nbcoeff(1)^2 - r^2) if r <= nbcoeff(1), or 0 otherwise

classical_mixrule (character*30)

This variable specifies the manner in which the parameters for unlike atoms are determined.

- 'Arithmetic': arithmetic mean of s.

- 'Repulsive Well': This is used to help setup and equilibrate a square well system where it is sometimes challenging to create an initial conformation with no overlaps. Use this option to equilibrate until the nonbonded potential energy is negative, and then switch back to the normal Square Well potential.

$U_{nonbond} = 1 \times 10^5 + 1 \times 10^5 * (\text{nbcoeff}(1)^2 - r^2)$ if r <= nbcoeff(1)

$U_{nonbond}$ = -nbcoeff(3) if nbcoeff(1) < r <= nbcoeff(2)

$U_{nonbond}$ = 0 if nbcoeff(2) < r

classical_mixrule (character*30)

This variable specifies the manner in which the parameters for unlike atoms are determined.

- 'Lorentz-Berthelot': Arithmetic mean for the hard core and square well diameters. Geometric mean for the well depth.
- 'Explicit': cross terms explicitly defined in the appropriate towhee_ff file.

- 'Scaled Lennard-Jones': 12-6 Lennard-Jones potential with scaling of coulombic and van der Waals terms for selected atoms. Typically used for thermodynamic integration (see Beutler *et al.* and Shirts *et al.*).

classical_mixrule (character*30)

This variable specifies the manner in which the parameters for unlike atoms are determined.

- 'Lorentz-Berthelot': arithmetic mean of s, geometric mean of e.
- 'Geometric': geometric mean of s and e.
- 'Explicit': cross terms explicitly specified in the appropriate towhee_ff file.
- 'Shukla': complicated expression involving the polarizability of each atom. See equations 19 and 20 of Shukla 1987 for the definition of this mixing rule.

cmix_rescaling_style (character*30)

This variable controls the rescaling of the Lennard-Jones and coulombic potentials, typically used in thermodynamic integration calculations. It is valid only for the "Scaled Lennard-Jones" **classical potential**.

- 'none': No additional adjustments to the nonbonded parameters as specified in the force field files and mixed according to the **classical_mixrule**. With this option, the potential is equivalent to the "Lennard-Jones" **classical potential**.
- 'soft-core': The Lennard-Jones and coulombic components of intermolecular interactions are scaled by λ_{LJ} and λ_C parameters, respectively, as described in Shirts *et al.* 2003.

The two-body potential between a scaled atom (as defined by **cmix_pair_list**, below) and an unscaled atom is given as,

$$U_{two\ body} = 4 * \lambda_{LJ}^4 * e * A (A - 1)$$

where

$$A^{-1} = a_{LJ} * (1-\lambda_{LJ})^2 + (r_{ij} / s)^6$$

For cases where neither, or both, atoms are scaled, the 'Lennard-Jones' classical potential is used. Note that for λ_{LJ}=1, this two-body term reduces to the 'Lennard-Jones' potential.

Coulombic interactions are also scaled for the same set of atoms. Given a coulombic potential $U_{coulomb}$ between a scaled atom (as defined by **cmix_pair_list**, below) and an unscaled atom, the scaled coulomb potential is given as,

$$U'_{coulomb} = \lambda_C U_{coulomb}$$

At present, only **coulombstyle** of 'minimum image' is supported.

The 'soft-core' **cmix_rescaling_style** option requires the following additional variables

cmix_lambda_lj (double precision)
: The Lennard-Jones scaling factor λ_{LJ}. Require 0 = **cmix_lambda_lj** = 1.

cmix_alpha_lj (double precision)
: The parameter a_{LJ}. Require **cmix_alpha_lj** = 0. Shirts *et al.* 2003 suggest a_{LJ}=0.5

cmix_lambda_c (double precision)
: The coulombic scaling factor λ_C. Require 0 = **cmix_lambda_c** = 1.

cmix_lprintdudl (logical)

- .TRUE. to evaluate and print $\partial U/\partial\lambda_{LJ}$ and $\partial U/\partial\lambda_C$, the derivatives of the internal energy with respect to the two lambda parameters.
- .FALSE. to ignore these calculations

cmix_npair (integer)
: The number of atom types that are modified by this rescaling style. Must be positive.

cmix_pair_list (character*10 array)
: The force field name and the atom type name for each value of **cmix_npair**. Listed with one force field name and atom type name per line.

lshift See the lshift section for information about this variable
rmin See the rmin section for information about this variable
rcut See the rcut section for information about this variable
rcutin See the rcutin section for information about this variable

---

- 'Stillinger-Weber': Stillinger-Weber potential (see Stillinger and Weber 1985)
  This is an atomic potential and can only be used with monatomic molecules in Towhee
  U = nbcoeff(1)*[nbcoeff(2)*Sum u2(r_{ij}) + nbcoeff(7)*Sum u3(r_{ij},r_{jk})]
  u2(rij) = [nbcoeff(3)*$\{r_{ij}$/nbcoeff(4)$\}^{-nbcoeff(5)}$ - 1] * exp{1/[(r_{ij}/nbcoeff(4) - nbcoeff(6))]} * Heaviside(nbcoeff(6) - [r_{ij}/nbcoeff(4)])
  u3(r_{ij},r_{jk}) = exp[nbcoeff(8)/{r_{ij}/nbcoeff(4) - nbcoeff(6)} + nbcoeff(8)/{r_{jk}/nbcoeff(4) - nbcoeff(6)}] * (cos($theta_{ijk}$)-nbcoeff(9))^2 * Heaviside(nbcoeff(6) - r_{ij}/nbcoeff(4)) * Heaviside(nbcoeff(6) - r_{jk}/nbcoeff(4))
  classical_mixrule (character*30)
  This variable specifies the manner in which the parameters for unlike atoms are determined.
  - 'Explicit': cross terms explicitly defined in the appropriate towhee_ff file.

- 'SW pair only': Stillinger-Weber potential (see Stillinger and Weber 1985) using only the pair potential
  This is an atomic potential and can only be used with monatomic molecules in Towhee. It is the same as the Stillinger-Weber potential but without the three-body terms that make it suitable for bonded materials. This option exists for academic study of the pair potential portion of the Stillinger-Weber functions. It reads from the same potential files as the 'Stillinger-Weber' potential.
  U = nbcoeff(1)*[nbcoeff(2)*Sum u2(r_{ij})]
  u2(rij) = [nbcoeff(3)*$\{r_{ij}$/nbcoeff(4)$\}^{-nbcoeff(5)}$ - 1] * exp{1/[(r_{ij}/nbcoeff(4) - nbcoeff(6))]} * Heaviside(nbcoeff(6) - [r_{ij}/nbcoeff(4)])
  classical_mixrule (character*30)
  This variable specifies the manner in which the parameters for unlike atoms are determined.
  - 'Explicit': cross terms explicitly defined in the appropriate towhee_ff file.

- 'Square Well': Square Well potential
  $U_{nonbond}$ = Infinity if r <= nbcoeff(1)
  $U_{nonbond}$ = -nbcoeff(3) if nbcoeff(1) < r <= nbcoeff(2)
  $U_{nonbond}$ = 0 if nbcoeff(2) < r
  classical_mixrule (character*30)

This variable specifies the manner in which the parameters for unlike atoms are determined.

- 'Lorentz-Berthelot': Arithmetic mean for the hard core and square well diameters. Geometric mean for the well depth.
- 'Explicit': cross terms explicitly defined in the appropriate towhee_ff file.

radial_pressure_delta See the radial_pressure_delta section for information about this variable

---

- 'Tabulated Pair': Tabulated Pair Potential

This potential uses a table to describe the interactions between atoms. The **interpolatestyle** determines the method for interpolating between the specified values of the pair potential terms. Cross terms between unlike atoms are described explicitly. The potential is listed with the distance and corresponding energy on each line.
interpolatestyle See the interpolatestyle section for information about this variable

---

- 'UFF 12-6': 12-6 potential with some additional nonbond coefficients.

This uses the same parameters as the Lennard-Jones for computing the nonbonded interactions
If the two atoms are separated by more than 3 bonds, or are on different molecules then
$U_{nonbond} = 4 * nbcoeff(2) * [(nbcoeff(1)/r)^{12} - (nbcoeff(1)/r)^{6}]$
else if the two atoms are separated by exactly 3 bonds then
$U_{nonbond} = 4 * nbcoeff(4) * [(nbcoeff(3)/r)^{12} - (nbcoeff(3)/r)^{6}]$
In addition, the following nbcoeff values are also stored. Some of these are required in order to compute the angle force constants.
nbcoeff(5): valence bond length (UFF r_I) in units of Angstroms.
nbcoeff(6): valence bond angle (UFF $theta_0$) in units of degrees.
nbcoeff(7): nonbond distance (UFF x_I) in units of Angstroms.
nbcoeff(8): nonbond energy (UFF D_I) in units of kcal/mol.
nbcoeff(9): nonbond scale factor (UFF zeta) in arbitrary units.
nbcoeff(10): effective charge (UFF Z_I) in charge units.
nbcoeff(11): electronegativity (UFF chi_I) in eV.
nbcoeff(12): torsion constant (UFF V_I) in units of kcal/mol.

classical_mixrule (character*30)

This variable specifies the manner in which the parameters for unlike atoms are determined.

- 'Geometric': geometric mean of s and e.

lshift See the lshift section for information about this variable
ltailc See the ltailc section for information about this variable
rmin See the rmin section for information about this variable
rcut See the rcut section for information about this variable
rcutin See the rcutin section for information about this variable

---

- 'Weeks-Chandler-Anderson': Repulsive-only version of 12-6 Lennard-Jones potential. See Weeks *et al.* 1970.

This potential truncates the LJ potential for distances greater than the LJ minimum. The well depth is shifted by e to make the remaining part all-repulsive. Useful for reference state calculations since the attractive part of LJ can be treated as a perturbation to the WCA repulsive-only potential.
classical_mixrule (character*30)

This variable specifies the manner in which the parameters for unlike atoms are determined.

- 'Lorentz-Berthelot': arithmetic mean of s, geometric mean of e.
- 'Geometric': geometric mean of s and e.
- 'Explicit': cross terms explicitly specified in the appropriate towhee_ff file.
- 'Shukla': complicated expression involving the polarizability of each atom. See equations 19 and 20 of Shukla 1987 for the definition of this mixing rule.

cmix_rescaling_style (character*30)

This variable controls the modification of the Lennard-Jones potential

- 'none': No additional adjustments to the nonbonded parameters as specified in the force field files and mixed according to the **classical_mixrule**. With this option, the potential is equivalent to the "Lennard-Jones" **classical potential**.

- 'WCA': Lennard-Jones interactions are modified as in Weeks *et al.* 1970.
  The potential between a modified atom (as defined by **cmix_pair_list** below) and any other atom less than $2^{1/6}$ * nbcoeff(1) away is:

  $$U_{nonbond} = 4 * nbcoeff(2) * [(nbcoeff(1)/r)^{12} - (nbcoeff(1)/r)^{6}] + nbcoeff(2)$$

  If the interatomic distance is greater than or equal to $2^{1/6}$ * nbcoeff(1) then:

  $$U_{nonbond} = 0$$

  For cases where neither atom is scaled, the 'Lennard-Jones' classical potential is used.
  This option requires the following additional variables
  - **cmix_npair (integer)**
    The number of atom types that are modified by this rescaling style. Must be positive.
  - **cmix_pair_list (character*10 array)**
    The force field name and the atom type name for each value of **cmix_npair**. Listed with one force field name and atom type name per line.
  - **lshift** See the lshift section for information about this variable
  - **ltailc** See the ltailc section for information about this variable
  - **rmin** See the rmin section for information about this variable
  - **rcut** See the rcut section for information about this variable
  - **rcutin** See the rcutin section for information about this variable

---

This section contains a description of variables that are identical in meaning regardless of the value of **classical_potential**. Please note that not all of these variables are required for each value of the **classical_potential** so you should consult the information above to determine which are required.

lshift (logical)
- .true. if you want the nonbonded potential to be shifted so that it is zero at the cutoff.
- .false. if you do not want to shift the nonbonded potential.

ltailc (logical)
- .true. if you want to apply analytical tail corrections for the portion of the potential that is past the cutoff. Note that you cannot have a shifted potential and tail corrections at the same time.
- .false. if you do not want analytical tail corrections.

rmin (double precision)
- A hard inner cutoff that can speed computation for Lennard-Jones systems, and is required to avoid the potential hitting infinity for exponential repulsion systems which also contain point charges. This should be set smaller than the smallest radius of any atom. The suggested default values are in the range from 0.5 to 1.0 Angstroms.

rcut (double precision)
- The distance (in Angstroms) beyond which the nonbonded potential (and density potential in the case of Embedded Atom Method) are no longer computed. This has no effect upon the coulombic potential as that is controlled by the **coulombstyle** variables. The **rcut** variable is also used to determine where to shift the potential (if **lshift** is true) or where to begin applying long range tail corrections (if **ltailc** is true). Note that this value is automatically adjusted upwards for certain types of Embedded Atom Method potentials.

rcutin (double precision)
- The inner nonbonded cutoff used in configurational-bias Monte Carlo moves. This dual-cutoff method can speed configurational-bias computations by at least a factor of 2, without affecting the acceptance rate. The inner cutoff is used during the growth procedure, and the full potential is calculated at the end of the move and everything is fixed up in the acceptance criteria. Suggested default values are 5 Angstroms for noncoulombic simulations, and 10 Angstroms for coulombic simulations.

interpolatestyle (character*20)
- 'cubicspline': Uses a cubic spline to interpolate between the tabulated portion of the force field data points provided in the force field files.
- 'linear': Linear interpolation between the data points of the tabulated portion of the force field

radial_pressure_delta (double precision)

This variable specifies the bin width that is used to compute local radial distribution functions near the discontinuities in the potential in order to calculate the pressure of the system. In general, setting this as close to zero as possible reduces the systematic errors of the method. However, as this setting gets very small the sampling becomes more difficult and the statistical errors become large. Please see the Towhee standard output manual for more information about the Radial pressure. The currently recommended default value is 1.0d-2.

Return to the main towhee web page

Send comments to: Marcus G. Martin
Last updated: October 10, 2007

MCCCS Towhee (towhee_input Database Feature)

Overview

This section covers the variables that are set in the towhee_input file when using the database feature. This manual is only updated for the current release version (last updated for Version 5.0.0). Each variable is listed along with its type (logical, character, integer, or double precision). towhee_input is the main input file for Towhee and is generally the only file that needs to be edited on a regular basis. It has a regimented style to the input. The variables are described here in the order they appear in this file.

Variable explanations for towhee_input

inputformat (character string)
- 'Database' : is the only valid option when utilizing the database features.

minimum vector length (double precision)
- The minimum length of any of the dimensions when creating the simulation box.

classical_potential (character*30)

The setting for this variable controls the nonbonded potential type. Depending on the setting there are a number of other variables that also must be listed. Please see the classical_potential web page for more information.

energyfitstyle (character*15)
- 'absolute': fits to the absolute energies specified in towhee_database. This means the binding energy of the solid is explicitly in the fit.
- 'relative': fits to the energies relative to the specified ground state. The binding energy of the ground state is not included in the fit and is instead computed as part of the fit.

loptimize (logical)
- .true. if you want to use least squares to fit the optimal prefactors for the linear combination of the strictly pair-potential and the multi-body parts of the forcefield to the data set.
- .false. if you want the answers without any fitting of parameters.

lprint (logical)
- .true. if you want to see the details of the energy output from engtotal for each of the data points in towhee_database. Please note that this is a lot of output and should definitely not be used in combination with the Dakota fitting procedure. It is intended for use when you desire more information about a single parameter set.
- .false. if you do not want extra energy information.

ffnumber (integer)
- 1 or more: reads the force field information from this number of file(s) listed in the **ff_filename**.

ff_filename (formatted character*70) [one line for each force field]
- A list of the filenames (one per line) that contain the force field information.

datapoints (integer)
- The number of datapoints in the training set to read from towhee_database.

scanstyle (character string)
- 'dakota' reads dakota_input and adjusts some force field parameters accordingly.
- 'none' no adjustment to the forcefield parameters.

The following are only required if **scanstyle** is 'dakota' and **classical_potential** is 'Embedded Atom Method'

nscale (integer)
- The number of interactions, involving pairs of atomtypes, that you wish to rescale.

interaction (character*5), atomone (character*2), atomtwo (character(2)
- List on **nscale** lines where each line contains the name of the interaction (embed or pair),atom name #1, and atom name #2

Return to the main towhee web page

Send comments to: Marcus G. Martin
Last updated: November 16, 2006

MCCCS Towhee (input_style explicit)

This section describes the input variables associated with an **input_style** setting of 'explicit'. Requires explicit declaration of all terms of the force field that describes this molecule type. This is most difficult way to set up the molecule information, but it also allows the most flexibility. This option is only recommended when it is not possible to use the more sophisticated options to set up the molecule of interest.

nunit (integer)

- The number of atoms (or united-atoms) in this molecule. Must be less than or equal to NUMAX (see preproc.h).

nmaxcbmc (integer)

- The maximum number of atoms to regrow during a configurational-bias regrowth.

lpdbnames (logical)

- .TRUE. : to enable the input of information about the pdb (protein data bank) atom name, amino acid number, amino acid name. This information is needed in order to use the cartoon feature of certain pdb viewers (such as Rasmol).
- .FALSE. : to disable the input of additional pdb information. This does not disable the output of pdb files, it just means thing like the Rasmol cartoon feature will not work properly.

The variables listed immediately below (unit through improper) are listed as a group for each atom in the molecule. These are repeated one time for each atom in the molecule. Thus, you input all of the information about the first atom before you list information for the subsequent atoms.
unit (integer), ntype (integer), qqatom (double precision)

- unit is the number of the atom in order starting from atom number 1. This is only used to help the user keep track of the molecule as they are building it in the input file. If the unit number listed in towhee_input does not match the running total of unit numbers in Towhee then the code will stop with an error message.
- ntype is the integer used to determine the force field parameters. You will need to look in the towhee_ff_xxx force field files you are utilizing in order to find the number that you wish to use.
- qqatom is the charge on this atom.

pdbname (character), aminonum (integer), aminoshort (character)
Note: These variables only need to be listed if lpdbnames = .true.

- pdbname: A four letter/number string that is output in the pdb file. The precise spacing is important if you want to get most pdb viewers to recognize the atoms as the pdb file is extremely specific.
- aminonum: The number of each amino acid starting from the N-terminus.
- aminoshort: The three letter code for each amino acid, or other group (such as caps on the C or N termini).

vibration

invib (integer)
- The first line under the vibration heading is the number of atoms that are bonded to the current atom. Must be a number between 0 and NNBOND (see preproc.h).

ijvib (integer), itvib (integer)
- The next **invib** lines underneath the vibration heading are a list of the bond partner and bond force field number for the **invib** atoms that are bonded to the current atom. Thus if you have 4 vibrations the next 4 lines will list the bond partner and bond force field number for each bond.

bending

inben (integer)

- The first line under the bending heading is the number of bond bending angles that terminate at the current atom. You must list all bond bending angles which have the current atom at one of the ends, but you do not list bond bending angles which contain the current atom in the center. Must be a number between 0 and MAXBEND (see preproc.h).

ijben2 (integer), ijben3(integer), itben (integer), orderben (integer)

- The next **inben** lines underneath the bending heading are a list of the other atoms in the bond bending angle and the bond bending angle force field number for the **inben** angles that contain the current atom at one of the ends. The format for listing the angle is to consider the current atom in the first position of an angle between atoms current-ijben2-ijben3 so you only need to list atoms ijben2 and ijben3, the bending type, and the bending order on each line, where there is one angle per line of towhee_input. The bending order is only used for force fields that have parameters that depend upon the order of the three atoms in the bond - typically those that have class 2 cross terms. Valid entries for the bending order are
  - 1 : the atoms in the bending angle force field are in the same order as the atoms listed here
  - -1 : the atoms in the bending angle force field are in the reverse order compared to the atoms listed here
  - 0 : null entry to make it easy for those force fields that do not depend upon the bending order

torsion

intor (integer)

- The first line under the torsion heading is the number of dihedral angles (regular torsions) that terminate at the current atom. You must list all regular torsion angles which have the current atom at one of the ends, but you do not list regular torsion angles which contain the current atom in the center. Must be a number between 0 and MAXTOR (see preproc.h).

ijtor2 (integer), ijtor3(integer), ijtor4 (integer), ittor (integer)

- The next **intor** lines underneath the torsion heading are a list of the other atoms in the regular torsion angle and the torsion force field number for the **intor** torsions that contain the current atom at one of the ends. The format for listing the regular torsion angle is to consider the current atom in the first position of a dihedral between atoms current-ijtor2-ijtor3-ijtor4 so you only need to list atoms ijtor2, ijtor3, and ijtor4 and the torsion type on each line, where there is one regular torsion per line of towhee_input.

angle-angle

inaa (integer)

- The first line under the angle-angle heading is the number of angle-angle terms which have their shared central atom located at the current atom. You must list all angle-angle terms which have the current atom at the shared central position, but you do not list angle-angle terms which contain the current atom at one of the ends. Must be a number between 0 and MAXAA (see preproc.h).

ijaa0 (integer), ijaa1(integer), ijaa2 (integer), itaa (integer)

- The next **inaa** lines underneath the angle-angle heading are a list of the other atoms in the angle-angle term and and the angle-angle force field number for the **inaa** angle-angle terms that contain the current atom at the shared central atom. The format for listing the regular torsion angle is to consider the current atom as the central shared atom in the angle-angle term between the angles ijaa0-current-ijaa1 and ijaa0-current-ijaa2. Each angle-angle term is listed on one line according to the format ijaa0, ijaa1, ijaa2 and itaa (the angle-angle type).

improper torsion

inimprop (integer)

- The first line under the improper heading is the number of improper torsions (any form) which have the central atom located at the current atom. You must list all improper torsions which have the current atom at the central position, but you do not list improper torsions which contain the current atom at one of the ends. Must be a number between 0 and MAXIMPROP (see preproc.h).

ijimprop2 (integer), ijimprop3(integer), ijimprop4 (integer), itimprop (integer)

- The next **inimprop** lines underneath the improper torsion heading are a list of the other atoms in the improper torsion and the improper force field number for the **inimprop** improper torsions that contain the current atom at the central atom. There are currently three different forms of improper torsions, and these forms are specified in the force field files. In all cases, the three atoms must be bound to the current atom.

This is the end of the section that is repeated for each atom for input_style 'explicit'
Return to the main towhee web page

Send comments to: Marcus G. Martin

Last updated: December 22, 2006

MCCCS Towhee (input_style 'polypeptide builder')

This section describes the input variables associated with an **input_style** setting of 'polypeptide builder'. Uses the polypeptide builder functionallity to set up everything for proteins from a list of the amino acids. Documentation of the currently available monomers is presented along with the forcefield information for each force field.

nunit (integer)

- The number of monomers in this molecule. Must be less than or equal to NUMAX (see preproc.h) and there is also a check in the builder to make sure you do not exceed NUMAX with the number of atoms that all of your monomers require.

nmaxcbmc (integer)

- The nmaxcbmc variable determines the maximum number of atoms that are regrown during a configurational-bias regrowth.

forcefield (character)

- The forcefield that you want to use to build this molecule.

protgeom (character)

- linear: The polypeptide will be generated starting from the N-terminus and finishing at the C-terminus. Also allows for multiple N and C terminii where the different linear polypeptides are connected by disulfide bonds.
- cyclic: The polypeptide is cyclic so the first amino acid is bonded to the final amino acid.

The variables listed immediately below are listed as a group for each amino acid in the molecule. There is only one line of input per building block in the molecule.
pepname (unformatted character string of length 2), stereochem (unformatted character string of length 1), bondpartner (integer), terminus (unformatted character of length 1)

- pepname is a two letter code for the building blocks of the molecule. Peptides use a code where the first letter is the one letter code for each amino acid. The second character is generally the charge state of the side chain, but has some different meanings for histidine and cysteine. The peptide codes can be found in the documentation for each force field./a>.
- stereochem is a single letter which determines the stereochemistry of the peptide. The valid options are l (L-amino acid), d (D-amino acid) and r (racemic).
- bondpartner is the number of the amino acid which is bonded to this amino acid through the side chain. This is only used for cysteines that have a disulfide bridge (type cs), otherwise set this value to 0. The bondpartner integers refer to the amino acids starting with the first amino acid listed in towhee_input as amino acid #1.
- terminus is a single character string which tells the code whether this peptide is at one of the terminii, or if it is in the central part of the polypeptide. You may have multiple N and C terminii, but all C terminii must be immediately preceded by an N-terminus.
    - 'N': N-terminus. All linear polypeptides must begin with an N-terminus
    - '-': Central polypeptide (anything that isn't an N or C terminus)
    - 'C': C-terminus. All linear polypeptides must end with a C-terminus.

This is the end of the section that is repeated for each atom for input_style 'polypeptide builder'
Return to the main towhee web page

Send comments to: Marcus G. Martin

Last updated: December 22, 2006

MCCCS Towhee (input_style 'basic connectivity map')

This section describes the input variables associated with an **input_style** setting of 'basic connectivity map'. Atom-based connectivity map, with the details of the force field parameters determined via the buildmolec and assemble routines. This documentation is only maintained for the current version of the code and was last modified for Version 5.1.0. This molecule assembly option was the standard way to set up any molecule that is not supported by one of the polymer builder options. It is still available as an option, although the advanced connectivity map option is suggested for most users as it provides a powerful and flexible version of this builder. The 'basic connectivity map' option requires the specification of a forcefield and the appropriate parameter names for that force field. The individual web pages for each of the force fields (summarized on the Towhee Capabilities page for more information about the atom names. This feature enables the automatic determination of all of the vibration types, bending angles, bending types, regular torsion angles, regular torsion types, angle-angle terms, angle-angle types, and improper torsion types that are implied by the bonding structure of the molecule. The automatic atom assignment for the improper torsions is not possible with this option and users who require that feature should use the advanced connectivity map option.

nunit (integer)

- The number of atoms (or united-atoms) in this molecule. Must be less than or equal to NUMAX (see preproc.h).

nmaxcbmc (integer)

- The maximum number of atoms to regrow during a configurational-bias regrowth move. It is normally best to set this to the same value as **nunit**, but if a user observes that no configurational-bias regrowth moves beyond a certain size are ever accepted then it is more efficient to set **nmaxcmbc** to the largest value where moves are accepted to avoid wasting effort on move that are so likely to fail.

lpdbnames (logical)

- .TRUE. if you want to input information about the pdb (protein data bank) atom name, amino acid number, amino acid name. This information is needed in order to use the cartoon feature of certain pdb viewers (such as Rasmol).
- .FALSE. if you don't need that feature. This does not disable the output of a pdb file, it just means things like the cartoon feature will not work properly in Rasmol.

forcefield (character*10)

- The forcefield that you want to use to build this molecule. A summary of available force fields is provided on the Towhee Capabilities page.

charge_assignment (character*30)

- 'bond increment': the bond increment method is used to assign the charges on each atom and the variable **qqatom** is not required. Note that this method is only functional for certain forcefields. Please check the individual force field documentation to see if this is available for your force field of interest.
- 'manual': the charges on each atom are manually specified in the towhee_input file as the variable **qqatom** in the following section.
- 'none': a zero charge is assigned to each atom and the variable **qqatom** is not required.

The variables listed immediately below (unit through improper) are listed as a group for each atom in the molecule. Thus, you input all of the information about the first atom before you list information for the subsequent atoms.

unit (integer), type (character), qqatom (double precision)

- **unit** is the number of the atom in order starting from atom number 1. This is only used to help the user keep track

of the molecule as they are building it in the input file. If the unit number listed in towhee_input does not match the running total of unit numbers in Towhee then the code will stop with an error message.
- **ntype** is the character string that contains the atom type for the forcefield. Information about valid atomtypes is found in the documentation for each force field (summarized on the Towhee Capabilities page).
- **qqatom** is the charge on this atom. This is only required if the **charge_assignment** variable is set to 'manual'.

pdbname (character), aminonum (integer), aminoshort (character)
Note: These variables only need to be listed if lpdb = .true.

- pdbname: A four letter/number string that is output in the pdb file. The precise spacing is important if you want to get most pdb viewers to recognize the atoms as the pdb file is extremely specific.
- aminonum: The number of each amino acid starting from the N-terminus.
- aminoshort: The three letter code for each amino acid, or other group (such as caps on the C or N termini).

vibration

invib (integer)
- The first line under the vibration heading is the number of atoms that are bonded to the current atom. Must be a number between 0 and NNBOND (see preproc.h).

ijvib (integer)
- The second line contains the atom numbers for all **invib** atoms that are bonded to the current atom.

improper torsion

inimprop (integer)
- The first line under the improper heading is the number of improper torsions (any form) which have the central atom located at the current atom. You must list all improper torsions which have the current atom at the central position, but you do not list improper torsions which contain the current atom at one of the ends. Must be a number between 0 and **MAXIMPROP** (see preproc.h).

ijimprop2 (integer), ijimprop3(integer), ijimprop4 (integer), itimprop (integer)
- The next **inimprop** lines underneath the improper torsion heading are a list of the other atoms in the improper torsion and the improper force field number for the **inimprop** improper torsions that contain the current atom at the central atom. There are currently three different forms of improper torsions, and these forms are specified in the force field. In all cases, the three atoms must all be bound to the current atom. If you want the program to determine the improper torsion type then just enter an itimprop value of 0.

This is the end of the section that is repeated for each atom for input_style 'basic connectivity map'

Return to the main towhee web page

Send comments to: Marcus G. Martin

Last updated: December 22, 2006

MCCCS Towhee (input_style 'nucleic acid builder')

This section describes the input variables associated with an **input_style** setting of 'nucleic acid builder'. Uses the nucleic acid builder functionallity to set up everything for DNA or RNA from a list of the nucleic acids. Documentation of the currently available monomers is presented along with the forcefield information for each force field.

nunit (integer)

- The number of monomers in this molecule. Must be less than or equal to NUMAX (see preproc.h) and there is also a check in the builder to make sure you do not exceed NUMAX with the number of atoms that all of your monomers require.

nmaxcbmc (integer)

- The nmaxcbmc variable determines the largest number of atoms that are regrown during a configurational-bias regrowth. Since we only know the number of amino acids at this stage, and not the number of atoms, the final value of **nmaxcbmc** is computed as the product of the final number of atoms in the molecule times the input value of **nmaxcbmc** divided by the input value of **nunit**.

terminus (integer)

- 3: the 3 prime end is terminated in a hydrogen and the 5 prime end has a phosphate group.
- 5: the 5 prime end is terminated in a hydrogen and the 3 prime end has a phosphate group.

forcefield (character)

- The forcefield that you want to use to build this molecule.

monomername (character)

- Two letter code for the building blocks of the molecule. The monomernames can be found in the documentation for the appropriate forcefields for use with nucleic acid simulations. The monomers are listed in order (one per line) starting from the 5 prime end.

Return to the main towhee web page

Send comments to: Marcus G. Martin

Last updated: December 22, 2006

MCCCS Towhee (input_style 'nanotube builder')

This section describes the input variables associated with an **input_style** setting of 'nanotube builder'. Uses the nanotube builder functionallity to set up the data structures and an initial conformation for carbon nanotubes from the traditional (m,n) nanotube chiral vector notation.

forcefield (character*10)

- The forcefield that you want to use to build this molecule.

atomname (character*6)

- Atom type for the building block of the nanotube. The atomnames can be found in the documentation for the forcefield of interest.

qqatom (double precision)

- The atomic charge on each atom of the nanotube.

nanotube_n (integer)

- The n index of the (n,m) chiral vector describing nanotube geometry.

nanotube_m (integer)

- The m index of the (n,m) chiral vector describing nanotube geometry.

nanotube_ncells (integer)

- The number of repeat cells to create for the nanotube. Must be positive.

nanotube_bondlength (double precision)

- The atom bond length used to create the initial structure.

Return to the main towhee web page

Send comments to: Marcus G. Martin

Last updated: Decemer 22, 2006

MCCCS Towhee (input_style 'atomic')

This section describes the input variables associated with an **input_style** setting of 'atomic'. This documentation is only maintained for the current version of the code. This molecule assembly option is required for use when using the 'Quantum' **classical_potential** and is very simple as quantum mechanical energies generally just use the element without requiring anything similar to classical atom types or a bonding pattern.

element (character*2)

- The name of the element using the standard two letter code from the periodic table.

quantum_gsenergy (double precision)

- The ground state energy of this element, expressed in units of Kelvin. This provides a constant shift in the reported energy and therefore has no effect on the acceptance rate of most of the moves, but allows the user to correct for the often quite large ground state energies of the atoms in order to output the energy on a sensible scale.

Return to the main towhee web page

Send comments to: Marcus G. Martin

Last updated: December 22, 2006

MCCCS Towhee (input_style 'advanced connectivity map')

This section describes the input variables associated with an **input_style** setting of 'advanced connectivity map'. This option adds additional flexibility compared to the similar basic connectivity map and is designed to replace that option. This documentation is only maintained for the current version of the code. This molecule assembly option is a powerful and relatively easy way to set up the molecular interactions by listing the atom types and providing their bond partners and rudimentary information about the bond orders. This option is designed to work with the force fields and atom names listed in the Towhee Capabilities section. This option determines all of the vibration types, bending angles, bending types, regular torsion angles, regular torsion types, angle-angle terms, angle-angle types, and improper torsion types that are implied by the bonding structure of the molecule. It also provides an option for a rule-based determination of improper torsions based upon the bonding pattern.

nunit (integer)

- The number of atoms (or united-atoms) in this molecule. Must be less than or equal to NUMAX (see preproc.h).

nmaxcbmc (integer)

- The maximum number of atoms to regrow during a configurational-bias regrowth. It is suggested to set this to the same value as **nunit**, but some molecules are so large that almost none of the moves which regrow large portions are accepted. In those cases setting a lower value of **nmaxcbmc** focused the software on moves that are likely to be accepted.

lpdbnames (logical)

- .TRUE. : to enable the input of information about the pdb (protein data bank) atom name, amino acid number, amino acid name. This information is needed in order to use the cartoon feature of certain pdb viewers (such as Rasmol).
- .FALSE. : to disable the input of additional pdb information. This does not disable the output of pdb files, it just means thing like the Rasmol cartoon feature will not work properly.

forcefield (character*10)

- The Towhee shorthand name for the forcefield that you want to use to build this molecule. See Towhee Capabilities for a complete list of the forcefields included with the Towhee distribution.

charge_assignment (character*30)

- 'bond increment': the bond increment method is used to assign the charges on each atom and the variable **charge** is not required. Note that this method is not functional for all forcefields.
- 'manual': the charges on each atom are manually specified in the towhee_input file as the variable **charge** in the following section.
- 'none': a zero charge is assigned to each atom and the variable **charge** is not required.

improper_assignment (character*30)

- '3-bond: single improper': atoms that are bonded to exactly three neighbors are assigned a single improper torsions automatically. The three neighbors are automatically assigned in increasing order of their nonbonded integer type number. Originally implemented for use with the MMFF94 force field. The improper torsion variables are not required in the subsequent section.
- 'manual': the improper torsions each atom are manually specified in the subsequent section.
- 'none': no improper torsions are assigned for this molecule and the improper torsion variables are not required in the subsequent section.

match_style (character*30)

- 'standard' : determines matching interactions for a given forcefield by checking for exact type matches first, and then utilizing a procedure that checks for an increasing number of wildcards. This is the approach recommended for the majority of forcefields and is the default value used by most of the other molecule assembly options.
- 'stereoselective' : very similar to the 'standard' option, but also requires the input of a list of torsions that have a specific stereochemistry. This allows the process to distinguish between cis and trans dihedrals when combined with a forcefield that makes such a distinction. When using this option the **special_torsion_count** variables are required after the usual input of the connectivity.
- 'MMFF checkdown' : determines matching interactions for a given forcefield using the procedure described in the MMFF94 forcefield papers. This uses an internal checkdown procedure based upon atom names that are implemented as a 10 character string that contains 5 2-character substrings.

The variables in this section are required for each atom in the molecule (determined by **nunit**), one atom at a time.
unit (integer), type (character), charge (double precision)

- **unit** is the number of the atom in order starting from atom number 1.
- **ntype** is the character string that contains the atom type for this atom. Information about valid atomtypes is found in individual forcefield documentation (see the Towhee Capabilities manual).
- **charge** is the charge on this atom. This is only required if the **charge_assignment** variable is set to 'manual'.

pdbname (character), aminonum (integer), aminoshort (character)

Note: These variables are only listed if **lpdbnames** = .TRUE.
- pdbname: A four character string that is output in the pdb file. The precise spacing is important if you want most pdb viewers to recognize the atoms as the pdb file is extremely specific.
- aminonum: The number of each amino acid starting from the N-terminus.
- aminoshort: The three letter code for each amino acid, or other group (such as caps on the C or N termini).

vibration

- The first line under the vibration heading is the number of atoms that are bonded to the current atom. Must be a number between 0 and NNBOND (see preproc.h).
- For each vibration a subsequent line contains the atom number (integer) of the bond partner and the vibration order (character string a10) of the bond. Commonly used vibration orders are listed here.
  - 'any': matches any bond order listed in the force field file. This effectively disables the vibration order searching. This entry is only valid in the towhee_input file, and not in an ff*.F file.
  - 'aromatic': aromatic bond
  - 'single': single bond
  - 'double': double bond
  - 'triple': triple bond
  - 'wild': matches with any type of bond, but only if there is not an exact match. This is only a valid entry in an ff*.F file, and not in the towhee_input file.
  -

improper torsion

Note: These variables are only listed for certain settings of **improper_assignment**
- The first line under the improper heading is the number of improper torsions (any form) that have the central atom located at the current atom. You must list all improper torsions that have the current atom at the central position, but you do not list improper torsions that only contain the current atom at one of the ends. Must be a number between 0 and **MAXIMPROP** (see preproc.h).
- For each improper torsion specified above a line with the following information is required. A list of the three

other atoms in the improper torsion followed by the appropriate improper type for the force field. The code currently only allows an improper torsion where the three other atoms are all bonded to the first atom. If you want to enable the automatic determination of the improper torsion force field parameters then set the improper force field type to 0.

This is the end of the section that is repeated for each atom

special_torsion_count

Note: These variables are only listed for certain settings of **match_style**

- The first line under the special_torsion_count heading is the number of special torsion assignments that involve a torsion order other than 'wild'. If this value is not 0, then a subsequent line is required for each value of this variable.
-
- For each value of the special_torsion_count specified above, a line with the following information is required. A list of the integer values of the 4 atoms that make up the special torsion, followed by a character string that describes the special torsion. The character string is checked against the torsion order specified in the various forcefield files in order to determine a match. The torsions only need to be specified in a single order, so if there was a special torsion between atoms 1,2,3, and 4 it is only required to list the 1 2 3 4 'special flag' direction and not the 4 3 2 1 'special flag' direction. The special torsion order character string depends upon the force field, but typical values are listed here.
    - 'wild' : default value if none is specified. This matches up with torsions that do not have a specified torsion order as the default output for the torsion order is 'wild'. There is no reason to go through the effort of listing this value since it is the default, but it would result in a valid match with the default torsions so it is allowed.
    - 'cis' : matches up with a torsion that is in the cis conformation only. Typically occurs for double bonded carbons that are then bonded to dissimilar atoms.
    - 'trans' : matches up with a torsion that is in the trans conformation only. Typically occurs for double bonded carbons that are then bonded to dissimilar atoms.

Return to the main towhee web page

Send comments to: Marcus G. Martin

Last updated: January 08, 2007

MCCCS Towhee (database_input_example_structure-energy)

Overview

This provides an example for the database_input option of "structure-energy".

structure-energy example

```
@structure
$start structure
ge_btin_m01gpa
$notes3
Ge b-tin/32^2x16/k12^2x24/LDA/SeqQ2.54/21Oct02, off-equil cell
ge_btin_m01gpa Etot= -31.91664380 Ry
E = E0 + 0.248 eV/atom
$distance unit
ANGSTROM
$dimension
3
$cell vectors
5.09922889 0.00000000 0.00000000
0.00000000 5.09922889 0.00000000
0.00000000 0.00000000 2.82817596
$number of atoms
4
$atom type position
1 Ge 0.0000000000 0.0000000000 0.0000000000
2 Ge 0.0000000000 2.5496144458 0.7070439907
3 Ge 2.5496144458 2.5496144458 1.4140879814
4 Ge 2.5496144458 0.0000000000 2.1211319721
$Ebinding cell
0.9910 eV
$Etotal cell
-31.916644 Ry
$end structure
```

Send comments to: Marcus G. Martin
Last updated: September 07, 2005

MCCCS Towhee Publications

Overview

This section contains references to papers and books utilizing the Towhee code since the project officially began in 1999. If you have published a paper using Towhee and wish to have it included here please contact Marcus G. Martin with the reference. The publications are arranged by year of publication and then alphabetical by author.

2006

- Y. Houndonougbo; J.-X. Guo; G. H. Lushington; B. Laird; "Monte Carlo simulations of CO_2-expanded acetonitrile", *Mol. Phys.* **104** 2955-2960 (2006).
- P. A. Gordon; "Development of intermolecular potentials for predicting transport properties of hydrocarbons", *J. Chem. Phys.* **125** 014504 (2006).
- M. G. Martin "Comparison of the AMBER, CHARMM, COMPASS, GROMOS, OPLS, TraPPE and UFF force fields for Prediction of vapor-liquid coexistence curves and liquid densities", *Fluid Phase Equilib.* **248** 50-55 (2006).
- M. G. Martin and A. L. Frischknecht, "Using arbitrary trial distributions to improve intramolecular sampling in configurational-bias Monte Carlo", *Mol. Phys.* **104** 2439-2456 (2006).
- D. Sabo, S. B. Rempe, J. A. Greathouse, and M. G. Martin, "Molecular studies of the structural properties of hydrogen gas in bulk water", *Mol. Sim.* **32** 269-278 (2006).
- A. O. Yazaydin; R. W. Thompson; "Simulating the vapour-liquid equilibria of 1,4-dioxane", *Mol. Sim.* **32** 657-662 (2006).

2005

- N. Du Preez; "Determination of Phase Equilibria for Long-chain Linear Hydrocarbons by Monte Carlo Simulation.", *University of KwaZulu/Natal Masters Thesis* (2005).
- L. J. D. Frink; M. Martin; "A combined molecular simulation-molecular theory method applied to a polyatomic molecule in dense solvent", *Condens. Matter Phys.* **8** 271-280 (2005).
- M. G. Martin; M. J. Biddy; "Monte Carlo Molecular Simulation Predictions for the Heat of Vaporization of Acetone and Butyramide", *Fluid Phase Equil.* **236** 53-57 (2005).

2004

- G. Kamath; F. Cao; J. J. Potoff; "An Improved Force Field for the Prediction of the Vapor-Liquid Equilibria of Carboxylic Acids", *J. Phys. Chem. B* **108** 14130-14136 (2004).
- M. G. Martin; A. P. Thompson; "Industrial property prediction using Towhee and LAMMPS", *Fluid Phase Equil.* **217** 105-110 (2004).

2003

- A. Narayanan; "Molecular Simulation of R-245fa with Pentane", *Tennessee Tech Masters Thesis* (2003).

2002

2001

- M. Chandross; E.B. Webb III; G.S. Grest; M.G. Martin; A.P. Thompson; M.W. Roth; "Dynamics of Exchange at Gas-Zeolite Interfaces I: Pure Component *n*-Butane and Isobutane", *J. Phys. Chem. B* **105** 5700-5712 (2001).
- M.G. Martin; A.P. Thompson; T.M. Nenoff; "Effect of pressure, membrane thickness, and placement of control volumes on the flux of methane through thin silicalite membranes: A dual control volume grand canonical molecular dynamics study", *J. Chem. Phys.* **114** 7174-7181 (2001).

Return to the main towhee web page

Send comments to: Marcus G. Martin

Last updated: September 19, 2007

MCCCS Towhee Presentations

Overview

This section contains references for public talks and posters utilizing the Towhee code since the project officially began in 1999. If you have presented a public talk or poster using Towhee and wish to have it included here please contact Marcus G. Martin with the reference. The presentations are arranged by year and then alphabetical by author.

2007

- P. Boulet, C. Vagner, D. Berger-Lefranc, O. Schaef, R. Denoyel and B. Kuchta, "Computational Chemistry Investigation of the Interaction of Paracresol with Alkali Cations and Water Within Zeolites. Application to Uremic Toxins Elimination", FOA9, 9th international conference on Fundamentals Of Adsorption, Giardini Naxos, Sicily, Italy. (May 20-25 2007).

2006

- Cessna M. Baca and Martha C. Mitchell, New Mexico State University, "Predicting Vapor-Liquid Equilibrium for Hydrazine using Gibbs Ensemble Monte Carlo Simulations with updated OPLS Force Field Parameters", The Eleventh Annual Undergraduate Research and Creative Arts Symposium, Las Cruces, NM (April 28, 2006).
- Marcus G. Martin, Sandia National Laboratories, "Recent advances in Configurational-bias Monte Carlo", 19th Annual Workshop: Recent Developments in Computer Simulation Studies in Condensed Matter Physics, University of Georgia, Athens, GA (February 2006).

2005

- Marcus G. Martin, Sandia National Laboratories, "Computing Vapor-Liquid Coexistence Curves for Metals", 2005 Annual AIChE Meeting, Cincinnati, OH (October 2005).
- Marcus G. Martin, Sandia National Laboratories, "Using the Monte Carlo Simulation Code Towhee to Predict Thermodynamic Properties of Fluids", AIChE Spring Meeting, Atlanta GA (April 11, 2005)

2004

- Nicholas du Preez, Marcus G. Martin Tyrone McKnight, Deresh Ramjugernath, University of Kwazulu-Natal and Sandia National Laboratories, "Pure Component Coexistence Properties for Long-chain 1-alkenes and 1-alcohols by Molecular Simulation Using Transferable Force Fields", ICCT 2004, Bejing China.
- Ganesh K Kamath, Jeffrey J Potoff, Wayne State University, "Phase Behavior and Structure of Acetic Acid.", 2004 AIChE Annual Meeting, Austin TX (October 2004).

2003

- Marcus G. Martin, Sandia National Laboratories, "Towhee: A Monte Carlo molecular simulation program for industrial applications", 226th ACS National Meeting, New York NY (September 2003).
- Jonathan Moore, Dow Chemical Company, "Predicting the Henry's Law Constants of Oxygen and Other Gases in Ethylene Oxide by Molecular Simulation", 2003 AIChE Annual Meeting, San Francisco CA (October 2003).
- Jonathan Moore, Dow Chemical Company, "Molecular Simulation for Industrial Physical Property Prediction", Colorado School of Mines, Golden CO (March 7, 2003).
- Jonathan Moore, Dow Chemical Company, "Prediction of gas solubility in ethylene oxide by molecular simulation", FOMMS 2003, Keystone CO (July 2003)

2002

- Marcus G. Martin, Sandia National Laboratories, "Improving the configurational-bias Monte Carlo algorithm for complex fluids", University of Minnesota, Minneapolis, MN (September 2002).
- Marcus G. Martin, Sandia National Laboratories, "Towhee: A Molecular Simulation Tool for Fluid Phase

Property Prediction", Wright-Patterson Air Force Base, Dayton, OH (August 2002).
- Marcus G. Martin, Sandia National Laboratories, "Towhee: A Molecular Simulation Tool for Fluid Phase Property Prediction", Dow Chemical, Midland, MI (April 2002).
- Marcus G. Martin, Sandia National Laboratories, "Molecular Simulation for the Chemical Industry", AIChE Spring Meeting, New Orleans, LA (March 2002).
- Jonathan Moore, Dow Chemical Company, "Application of theory and simulation to industrial problems in phase equilibria and interfacial modification", 224th ACS National Meeting, (August 22, 2002).
- Jonathan Moore, Dow Chemical Company, "Evaluating Molecular Simulation for Industrial Physical Property Prediction", 2002 AIChE Annual Meeting (November 8, 2002).

2001

- Marcus G. Martin, Laura J.D. Frink, Sandia National Laboratories, "Using Configurational-bias Monte Carlo to Perform Flexible Drug Docking", 2001 AIChE Annual Meeting, Reno, NV (11/2001).

2000

- Marcus G. Martin, Sandia National Laboratories, "Comparison of the Amber, Charmm, Compass, Gromos, OPLS and TraPPE Force Fields", 2000 AIChE Annual Meeting, Los Angeles, CA (11/2000).
- Marcus G. Martin, Sandia National Laboratories, "Assessing the accuracy of biological force fields for molecular simulation", Rush University, Chicago, IL (10/2000).

Return to the main towhee web page

Send comments to: Marcus G. Martin

Last updated: September 18, 2007

MCCCS Towhee (references)

Overview

This section contains references to papers and books that are either the source materials for the methods or force fields in Towhee, or are helpful reviews of those subjects. The references are arranged by subject, date of publication, and then alphabetical by author.

The Towhee Code

These are the currently suggested references for the Towhee code. Efforts are underway to prepare a review article that will serve as the "official" Towhee reference.

- http://towhee.sourceforge.net
- M. G. Martin; J. I. Siepmann; "Novel configurational-bias Monte Carlo method for branched molecules. Transferable potentials for phase equilibria. 2. united-atom description of branched alkanes", *J. Phys. Chem. B* **103** 4508-4517 (1999).

Experimental Data

- B. D. Smith; R. Srivastava; "Thermodynamic Data for Pure Compounds: Part A Hydrocarbons and Ketones.", Elsevier: Amsterdam (1986).
- B. D. Smith; R. Srivastava; "Thermodynamic Data for Pure Compounds: Part B Halogenated hydrocarbons and alcohols.", Elsevier: Amsterdam (1986).
- F. Hensel; W. W. Warren Jr.; "Fluid Metals", Princeton University Press: New Jersey (1999).

Force Fields: Ackland *et al.* 2004

- G. J. Ackland; M. I. Mendelev; D. J. Srolovitz; S. Han; A. V. Barashev; "Development of an interatomic potential for phosphorus impurities in alpha-iron", *J. Phys.: Condens. Matter* **16** S2629-S2642 (2004).

Force Fields: Alavi *et al.* 2005

- S. Alavi; J. A. Ripmeester; D. D. Klug; "Molecular-dynamics study of structure II hydrogen clathrates", *J. Chem. Phys.* **123** 024507 (2005).

Force Fields: Amber

- S. J. Weiner; P. A. Kollman; D. A. Case; U. C. Singh; C. Ghio; G. Alagona; S. Profeta Jr.; P. Weiner; "A New Force Field for Molecular Mechanical Simulation of Nucleic Acids and Proteins", *J. Am. Chem. Soc.* **106** 765-784 (1984).
- S. J. Weiner; P. A. Kollman; D. T. Nguyen; D. A. Case; "An All Atom Force Field for Simulations of Proteins and Nucleic Acids", *J. Comp. Chem.* **7** 230-252 (1986).
- J. J. Dick; J. P. Ritchie; "Molecular mechanics modeling of shear and the crystal orientation dependence of the elastic precursor shock strength in pentaerythritol tetranitrate", *J. Appl. Phys.* **76** 2726-2737 (1994).
- W. D. Cornell; P. Cieplak; C. I. Bayly; I. R. Gould; K. M. Merz Jr.; D. M. Ferguson; D. C. Spellmeyer; T. Fox; J. W. Caldwell; P. A. Kollman; "A Second Generation Force Field for the Simulation of Proteins, Nucleic Acids, and Organic Molecules", *J. Am. Chem. Soc.* **117** 5179-5197 (1995).

Force Fields: Aqvist Ions

- J. Aqvist; "Ion-Water interaction Potentials Derived from Free Energy Perturbation Simulations", *J. Phys. Chem.* **94** 8021-8024 (1990).

Force Fields: Catlow/Faux

- R. A. Jackson; C. R. A. Catlow; "Computer Simulation Studies of Zeolite Structure", *Mol. Sim.* **1** 207-224 (1988).
- C. R. A. Catlow; C. M. Freeman; B. Vessal; S. M. Tomlinson; M. Leslie; "Molecular Dynamics Studies of

Hydrocarbon Diffusion in Zeolites", *J. Chem. Soc. Faraday Trans.* **87** 1947-1950 (1991).

- D. A. Faux, W. Smith; T. R. Forester; "Molecular Dynamics Studies of Hydrated and Dehydrated Na^+-Zeolite-4A", *J. Phys. Chem. B* **101** 1762-1768 (1997).
- N. Raj; G. Sastre; C. R. A. Catlow; "Diffusion of Octane in Silicatlite: A Molecular Dynamics Study", *J. Phys. Chem. B* **103** 11007-11015 (1999).

Force Fields: Charmm

- E. Neria; S. Fischer; M. Karplus; "Simulation of Activation Free Energies in Molecular Systems", *J. Chem. Phys.* **105** 1902-1921 (1996).
- A. D. MacKerell Jr.; D. Bashford; M. Bellott; R. L. Dunbrack Jr.; J. D. Evanseck; M. J. Field; S. Fischer; J. Gao; H. Guo; S. Ha; D. Joseph-McCarthy; L. Kuchnir; K. Kuczera; F. T. K. Lau; C. Mattos; S. Michnick; T. Ngo; D. T. Nguyen; B. Prodhom; W. E. Reiher III; B. Roux; M. Schlenkrich; J. C. Smith; R. Stote; J. Straub; M. Watanabe; J. Wiorkiewicz-Kuczera; D. Yin; M. Karplus; "All-Atom Empirical Potential for Molecular Modeling and Dynamics Studies of Proteins", *J. Phys. Chem. B* **102** 3586-3616 (1998).
- T. Lazaridis; M. Karplus; "Effective Energy Function for Proteins in Solution", *Proteins* **35** 133-152 (1999).
- N. Foloppe; A. D. MacKerell Jr.; "All-Atom Empirical Force Field for Nucleic Acids: I. Parameter Optimization Based on Small Molecule and Condensed Phase Macromolecular Target Data", *J. Comp. Chem.* **21** 86-104 (2000).
- I. J. Chen; D. Yin; A. D. MacKerell Jr.; "Combined Ab initio/Empirical approach for Optimization of Lennard-Jones Parameters for Polar-Neutral Compounds", *J. Comp. Chem.* **23** 199-213 (2001).

Force Fields: ClayFF

- R. T. Cygan; J.-J. Liang; A. G. Kalinichev; "Molecular Models of Hydroxide, Oxyhydroxide, and Clay Phases and the Development of a General Force Field", *J. Phys. Chem. B* **108** 1255-1266 (2004).

Force Fields: Compass

- D. Rigby; H. Sun; B. E. Eichinger; "Computer Simulations of Poly(ethylene oxide): Force Field, PVT Diagram and Cyclization Behaviour", *Polymer International* **44** 311-330 (1997).
- H. Sun; D. Rigby; "Polysilxanes: ab initio force field and structural, conformational and thermophysical properties", *Spectrochimica Acta Part A* **53** 1301-1323 (1997).
- H. Sun; "COMPASS: An ab Initio Force-Field Optimized for Condensed-Phase Applications-Overview with Details on Alkane and Benzene Compounds", *J. Phys. Chem. B* **102** 7338-7364 (1998).
- H. Sun; P. Ren; J. R. Fried; "The COMPASS force field: parameterization and validation for phosphazenes", *Computational and Theoretical Polymer Science* **8** 229-246 (1998).
- S. W. Bunte; H. Sun; "Molecular Modeling of Energetic Materials: The Parameterization and Validation of Nitrate Esters in the COMPASS Force Field", *J. Phys. Chem. B* **104** 2477-2489 (2000).
- J. Yang; Y. Ren; A. Tian; H. Sun; "COMPASS Force Field for 14 Inorganic Molecules, He, Ne, Ar, Kr, Xe, H_2, O_2, N_2, NO, CO, CO_2, NO_2, CS_2, and SO_2, in Liquid Phases", *J. Phys. Chem. B* **104** 4951-4957 (2000).
- M. J. McQuaid; H. Sun; D. Rigby; "Development and Validation of COMPASS Force Field Parameters for Molecules with Aliphatic Azide Chains", *J. Comp. Chem.* **25** 61-71 (2003).

Force Fields: Coon1987

- J. E. Coon; S. Gupta; E. McLaughlin; "Isothermal-Isobaric Molecular Dynamics Simulation of Diatomic Liquids and Their Mixtures", *Chem. Phys.* **113** 43-52 (1987).

Force Fields: Cui1998

- S. T. Cui; J. I. Siepmann; H. D. Cochran; P. T. Cummings; "Intermolecular potentials and vapor-liquid phase equilibria of perfluorinated alkanes", *Fluid Phase Equilib.* **146** 51-61 (1998).

Force Fields: Cui2002

- J. Cui; J. R. Elliott Jr.; "Phase diagrams for a multistep potential model of *n*-alkanes by discontinuous molecular dynamics and thermodynamic perturbation theory", *J. Chem. Phys.* **116** 8625-8631 (2002).

Force Fields: Dacnis

- M. G. Martin; A. P. Thompson; T. M. Nenoff; "Effect of pressure, membrane thickness, and placement of control volumes on the flux of methane through thin silicalite membranes: A dual control volume grand canonical molecular dynamics study", *J. Chem. Phys.* **114** 7174-7181 (2001).

Force Fields: DREIDING

- S. L. Mayo; B. D. Olafson; W. A. Goddard III; "DREIDING: A Generic Force Field for Molecular Simulations", *J. Phys. Chem.* **94** 8897-8909 (1990).

Force Fields: Dubb2004

- D. Dubbeldam; S. Calero; T. J. H. Vlugt; R. Krishna; T. L. M. Maesen; E. Beerdsen; B. Smit; "Force Field Parametrization through Fitting on Inflection Points in Isotherms", *Phys. Rev. Lett.* **93** (2004).
- D. Dubbeldam; S. Calero; T. J. H. Vlugt; R. Krishna; T. L. M. Maesen; B. Smit; "United Atom Force Field for Alkanes in Nanoporous Materials", *J. Phys. Chem. B* **108** 12301-12313 (2004).
- S. Calero; D. Dubbeldam; R. Krishna; B. Smit; T. J. H. Vlugt; J. F. M. Denayer; J. A. Martens; T. L. M. Maesen; "Understanding the Role of Sodium during Adsorption: A Force Field for Alkanes in Sodium-Exchanged Faujasites", *J. Am. Chem. Soc.* **126** 11377-11386 (2004).

Force Fields: Embedded Atom Method

- M. S. Daw; M. I. Baskes; "Semiemperical, Quantum Mechanical Calculation of Hydrogen Embrittlement in Metals", *Phys. Rev. Lett.* **50** 1285-1288 (1983).
- S. M. Foiles; M. I. Baskes; M. S. Daw; "Embedded-atom-method functions for the fcc metals Cu, Ag, Au, Ni, Pd, Pt, and their alloys", *Phys. Rev. B* **33** 7983-7991 (1986).
- H. S. Lim; C. K. Ong; F. Ercolessi; "Stability of face-centered cubic and icosahedral lead clusters", *Surface Science* **269/270** 1109-1115 (1992).
- J. J. Hoyt; J. W. Garvin; E. B. Webb III; M. Asta; "An embedded atom method interatomic potential for the Cu-Pb system", *Modelling Simul. Mater. Sci. Eng.* **11** 1-13 (2003).

Force Fields: Elementary Physical Model (EPM)

- J. G. Harris; K. H. Yung; "Carbon Dioxide's Liquid-Vapor Coexistence Curve and Critical Properties As Predicted by a Simple Molecular Model", *J. Phys. Chem.* **99** 12021-12024 (1995).

Force Fields: Elliott 2002 (Elli2002)

- J. R. Elliott Jr.; "Optimized step potential models for *n*-alkanes and benzene", *Fluid Phase Equilib.* **194-197** 161-168 (2002).

Force Fields: Frischknecht and Curro 2003

- A. L. Frischknecht; J. G. Curro; "Improved United Atom Force Field for Poly(dimethylsiloxane)", *Macromolecules* **36** 2122-2129 (2003).

Force Fields: Galassi and Tildesley 1994

- G. Galassi; D. J. Tildesley; "Phase Diagrams of Diatomic Molecules Using the Gibbs Ensemble Monte Carlo Method", *Mol. Sim.* **13** 11-24 (1994).

Force Fields: Gromos

- W. F. van Gunsteren; S. R. Billeter; A. A. Eising; P. H. Hunenberger; P. Kruger; A. E. Mark; W. R. P. Scott; I. G. Tironi; "Biomolecular Simulation: The GROMOS96 Manual and User Guide.", VdF: Hochschulverlag AG an der ETH Zurich and BIOMOS b.v, Zurich, Gronigen (1996). ISBN 3 7281 2422 2.
- W. R. P. Scott; P. H. Hunenberger; I. G. Tironi; A. E. Mark; S. R. Billeter; J. Fennen; A. E. Torda; T. Huber; P. Kruger; W. F. van Gunsteren; "The GROMOS Biomolecular Simulation Program Package", *J. Phys. Chem. A* **103** 3596-3607 (1999).

Force Fields: Jaramillo

- E. Jaramillo; C. P. Grey; S. M. Auerbach; "Molecular Dynamics Studies of Hydrofluorocarbons in Faujasite-type Zeolites: Modeling Guest-Induced Cation Migration in Dry Zeolites", *J. of Phys. Chem. B.* **105** 12319-12329 (2001).

Force Fields: Kramer-Farragher-van Beest-van Santen

- B. W. H. van Beest; G. J. Kramer; R. A. van Santen; "Force Fields for Silicas and Aluminophosphates Based on *Ab Initio* Calculations", *Phys. Rev. Lett.* **64** 1955-1958 (1990).
- G. J. Kramer; N. P. Farragher; B. W. H. van Beest; R. A. van Santen; "Interatomic force fields for silicas, aluminophosphates, and zeolites: Derivation based on *ab initio* calculations", *Phys. Rev. B* **43** 5068-5080 (1991).

Force Fields: Lybrand-Ghosh-McCammon Ions

- T. P. Lybrand; I. Ghosh; J. A. McCammon; "Hydration of Chloride and Bromide Anions: Determination of Relative Free Energy by Computer Simulation", *J. Am. Chem. Soc.* **107** 7793-7794 (1985).

Force Fields: Mendelev *et al.* 2003 Iron

- M. I. Mendelev; S. Han; D. J. Srolovitz; G. J. Ackland; D. Y. Sun; M. Asta; "Development of new interactomic potentials appropriate for crystalline and liquid iron", *Phil. Mag.* **83** 3977-3994 (2003).

Force Fields: MCY1976

- O. Matsuoka; E. Clementi; M. Yoshimine; "CI study of the water dimer potential surface", *J. Chem. Phys.* **64** 1351-1361 (1976).

Force Fields: MM2

- N. L. Allinger; "Conformational Analysis. 130. MM2. A Hydrocarbon Force Field Utilizing V_1 and V_2 Torsional Terms", *J. Am. Chem. Soc.* **99** 8127-8134 (1977).

Force Fields: MMFF94

- T. A. Halgren; "Merck Molecular Force Field. I. Basis, Form, Scope, Parameterization, and Performance of MMFF94", *J. Comp. Chem.* **5 & 6** 490-519 (1996).
- T. A. Halgren; "Merck Molecular Force Field. II. MMFF94 van der Waals and Electrostatic Parameters for Intermolecular Interactions", *J. Comp. Chem.* **5 & 6** 520-552 (1996).
- T. A. Halgren; "Merck Molecular Force Field. III. Molecular Geometries and Vibrational Frequencies for MMFF94", *J. Comp. Chem.* **5 & 6** 553-586 (1996).
- T. A. Halgren; R. B. Nachbar; "Merck Molecular Force Field. IV. Conformational Energies and Geometries for MMFF94", *J. Comp. Chem.* **5 & 6** 587-615 (1996).
- T. A. Halgren; "Merck Molecular Force Field. V. Extension of MMFF94 Using Experimental Data, Additional Computational Data, and Empirical Rules", *J. Comp. Chem.* **5 & 6** 616-641 (1996).
- T. A. Halgren; "MMFF VII. Characterization of MMFF94, MMFF94s, and Other Widely Available Force Fields for Conformational Energies and for Intermolecular-Interaction Energies and Geometries", *J. Comp. Chem.* **7** 730-748 (1999).

Force Fields: Morrow and Maginn 2002

- T. I. Morrow; E. J. Maginn; "Molecular Dynamics Study of the Ionic Liquid 1-*n*-Butyl-3-methylimidazolium Hexafluorophosphate", *J. Phys. Chem. B* **106** 12807-12813 (2002).
- T. I. Morrow; E. J. Maginn; "2002, Vol. 106B", *J. Phys. Chem. B* **107** 9160-9160 (2003).

Force Fields: NERD (all versions)

- S. K. Nath; F. A. Escobedo; J. J. de Pablo; "On the simulation of vapor-liquid equilibria for alkanes", *J. Chem. Phys.* **108** 9905-9911 (1998).
- S. K. Nath; J. J. de Pablo; "Simulation of vapour-liquid equilibria for branched alkanes", *Mol. Phys.* **98** 231-238 (2000).
- S. K. Nath; B. J. Banaszak; J. J. de Pablo; "A new united atom force field for alpha-olefins", *J. Chem. Phys.* **114** 3612-3616 (2001).
- S. K. Nath; R. Khare; "New forcefield parameters for branched hydrocarbons", *J. Chem. Phys.* **115** 10837-10844 (2001).
- S. K. Nath; "Molecular Simulation of Vapor-Liquid Phase Equilibria of Hydrogen Sulfide and Its Mixtures with Alkanes", *J. Phys. Chem. B* **107** 9498-9504 (2003).
- R. Khare; A. K. Sum; S. K. Nath; J. J. de Pablo; "Simulation of Vapor-Liquid Phase Equilibria of Primary Alcohols and Alchol-Alkane Mixtures", *J. Phys. Chem. B* **108** 10071-10076 (2004).

Force Fields: OPLS and TIP*P

- W. L. Jorgensen; J. Chandrasekhar; J. D. Madura; R. W. Impey; M. L. Klein; "Comparison of simple potential functions for simulating liquid water", *J. Chem. Phys.* **79** 926-935 (1983).
- J. Chandrasekhar; D. Spellmeyer; W. L. Jorgensen; "Energy component analysis for dilute aqueous solutions of Li+, Na+, F- and Cl- Ions", *J. Am. Chem. Soc.* **106** 903-910 (1984).
- M. E. Cournoyer; W. L. Jorgensen; "Solvent Effects on the Relative Energies of Carbonium Ions. Solvation and Internal Rotation for the Allyl Cation in Liquid Hydrogen Fluoride", *J. Am. Chem. Soc.* **106** 5104-5112 (1984).
- W. L. Jorgensen; J. D. Madura; C. J. Swenson; "Optimized Intermolecular Potential Functions for Liquid Hydrocarbons", *J. Am. Chem. Soc.* **106** 6638-6646 (1984).
- W. L. Jorgensen; C. J. Swenson; "Optimized Intermolecular Potential Functions for Amides and Peptides. Structure and Properties of Liquid Amides", *J. Am. Chem. Soc.* **107** 569-578 (1985).
- W. L. Jorgensen; "Optimized Intermolecular Potential Functions for Liquid Alcohols", *J. Phys. Chem.* **90** 1276-1284 (1986).
- W. L. Jorgensen; "Intermolecular Potential Functions and Monte Carlo Simulations for Liquid Sulfur Compounds", *J. Phys. Chem.* **90** 6379-6388 (1986).
- W. L. Jorgensen; J. M. Briggs; "Monte Carlo simulations of liquid acetonitrile with a three-site model", *Mol. Phys.* **63** 547-558 (1988).
- W. L. Jorgensen; J. M. Briggs; M. L. Contreras; "Relative Partition Coefficients for Organic Solutes from Fluid Simulations", *J. Phys. Chem.* **94** 1683-1686 (1990).
- J. Pranata; S. G. Wierschke; W. L. Jorgensen; "OPLS potential functions for nucleotide bases. Relative association constants of hydrogen-bonded base pairs in chloroform", *J. Am. Chem. Soc.* **113** 2810-2819 (1991).
- G. Kaminski; E. M. Duffy; T. Matsui; W. L. Jorgensen; "Free energies of hydration and pure liquid properties of hydrocarbons from the OPLS all-atom model", *J. Phys. Chem.* **98** 13077-13082 (1994).
- W. L. Jorgensen; D. S. Maxwell; J. Tirado-Rives; "Development and testing of the OPLS all-atom force field on conformational energetics and properties of organic liquids", *J. Am. Chem. Soc.* **118** 11225-11236 (1996). {see the supporting information}
- W. L. Jorgensen; "OPLS all-atom parameters for organic molecules, ions, & nucleic acids 12/96", available by sending an email to W. L. Jorgensen.
- W. Damm; A. Frontera; J. Tirado-Rives; W. L. Jorgensen; "OPLS all-atom force field for carbohydrates", *J. Comp. Chem.* **18** 1955-1970 (1997).
- W. L. Jorgensen; N. A. McDonald; "Development of an all-atom force field for heterocycles. Properties of liquid pyridine and diazenes", *J. Mol. Structure Theochem* **424** 145-155 (1998).
- N. A. McDonald; W. L. Jorgensen; "Development of an all-atom force field for heterocycles. Properties of liquid

pyrrole, furan, diazoles, and oxazoles", *J. Phys. Chem. B* **102** 8049-8059 (1998).

- N. A. McDonald; E. M. Duffy; W. L. Jorgensen; "Monte Carlo investigations of selective anion complexation by a bis(phenylurea)p-tert-butylcalix[4]arene", *J. Am. Chem. Soc.* **120**, 5104-5111 (1998).
- R. C. Rizzo; W. L. Jorgensen; "OPLS all-atom model for amines: resolution of the amine hydration problem", *J. Am. Chem. Soc.* **121** 4827-4836 (1999).
- M. W. Mahoney; W. L. Jorgensen; "A five-site model for liquid water and the reproduction of the density anomoly by rigid, nonpolarizable potential functions", *J. Chem. Phys.* **112** 8910-8922 (2000).
- W. L. Jorgensen; "OPLS all-atom parameters for organic molecules, ions, & nucleic acids 5/01", available by sending an email to W. L. Jorgensen.
- G. A. Kaminski; R. A. Friesner; J. Tirado-Rives; W. L. Jorgensen; "Evaluation and Reparametrization of the OPLS-AA Force Field for Proteins via Comparison with Accurate Quantum Chemical Calculations on Peptides", *J. Phys. Chem. B* **105** 6474-6487 (2001).

Force Fields: Panagiotopoulos 1989 noble gasses

- C. G. Gray; K. E. Gubbins; "Theory of Molecular Fluids Volume 1: Fundamentals" *Clarendon Press* **Oxford** (1984).
- A. Z. Panagiotopoulos; "Exact Calculations of Fluid-Phase Equilibria by Monte Carlo Simulation in a New Statistical Ensemble", *Int. J. Thermophys.* **10** 447-457 (1989).

Force Fields: Potter *et al.* 1997

- S. C. Potter; D. J. Tildesley; A. N. Burgess; S. C. Rogers; "A transferable potential model for the liquid-vapour equilibria of fluoromethanes", *Mol. Phys.* **92** 825-833 (1997).

Force Fields: QMFF-VIII

- C. S. Ewig; R. Berry; U. Dinur; J.-R. Hill; M.-J. Hwang; H. Li; C. Liang; J. Maple; Z. Peng; T. P. Stockfisch; T. S. Thacher; L. Yan; X. Ni; A. T. Hagler; "Derivation of Class II Force Fields. VIII. Derivation of a General Quantum Mechanical Force Field for Organic Compounds", *J. Comp. Chem.* **22** 1782-1800 (2001).

Force Fields: Richards *et al.* 1995

- A. J. Richards; K. Watanabe; N. Austin; M. R. Stapleton; "Computer Simulation of the Gas Separation Properties of Zeolite Li-X", *J. Porous Mater.* **2** 43-49 (1995).

Force Fields: Shah and Maginn 2002

- J. K. Shah; E. J. Maginn; "A Monte Carlo simulation study of the ionic liquid 1-*n*-butyl-3-methylimidazolium hexafluorophosphate: liquid structure, volumetric properties and infinite dilution solution thermodynamics of CO_2" *Fluid Phase Equilib.* **222-223** 195-203 (2004).

Force Fields: Shukla 1987

- K. P. Shukla; "Thermodynamic properties of simple fluid mixtures from perturbation theory", *Mol. Phys.* **62** 1143-1163 (1987).

Force Fields: SKS n-alkanes

- B. Smit; S. Karaborni; J. I. Siepmann; "Computer simulations of vapor-liquid phase equilibria of *n*-alkanes", *J. Chem. Phys.* **102** 2126-2140 (1995).
- B. Smit; S. Karaborni; J. I. Siepmann; "Erratum: Computer simulations of vapor-liquid phase equilibria of *n*-alkanes [J. Chem. Phys. 102, 2126 (1995)]", *J. Chem. Phys.* **109** (1998).

Force Fields: Simple Point Charge - Extended

H. J. C. Berendsen; J. R. Grigera; T. P. Straatsma; "The Missing Term in Effective Pair Potentials", *J. Phys. Chem.* **91** 6269-6271 (1987).

Force Fields: SMMK

- C. J. Mundy; S. Balasubramanian; K. Bagchi; J. I. Siepmann; M. L. Klein; "Equilibrium and non-equilibrium simulation studies of fluid alkanes in bulk and at interfaces", *Faraday Discuss.* **104** 17-36 (1996).
- J. I. Siepmann; M. G. Martin; C. J. Mundy; M. L. Klein; "Intermolecular potentials for branched alkanes and the vapour-liquid phase equilibria of n-heptane, 2-methylhexane, and 3-ethylpentane", *Mol. Phys.* **90** 687-693 (1997).

Force Fields: Smith and Dang 1994

- D. E. Smith; L. X. Dang; "Computer simulations of NaCl association in polarizable water", *J. Chem. Phys.* **100** 3757-3766 (1994).

Force Fields: Stillinger-Weber Potential

- F. H. Stillinger; T. A. Weber; "Computer simulation of local order in condensed phases of silicon", *Phys. Rev. B* **31** 5262-5271 (1985).
- K. Ding; H. C. Andersen; "Molecular-dynamics simulation of amorphous germanium", *Phys. Rev. B* **34** 6987-6991 (1986).
- R. L. C. Vink; G. T. Barkema; W. F. van der Weg; N. Mousseau; "Fitting the Stillinger-Weber potential to amorphous silicon", *Journal of Non-crystalline Solids* **282** 248-255 (2001).

Force Fields: Sum *et al.* 2003

- A. K. Sum; M. J. Biddy; J. J. de Pablo; M. J. Tupy; "Predictive Molecular Model for the Thermodynamic and Transport Properties of Triacylglycerols", *J. Phys. Chem. B* **107** 14443-14451 (2003).

Force Fields: Teleman *et al.* 1987

- O. Teleman; B. Jonsson; S. Engstrom; "A molecular dynamics simulation of a water model with intramolecular degrees of freedom", *Mol. Phys.* **60** 193-203 (1987).

Force Fields: TraPPE

- M. G. Martin; J. I. Siepmann; "Transferable potentials for phase equilibria. 1. United-atom description of n-alkanes", *J. Phys. Chem. B* **102** 2569-2577 (1998).
- M. G. Martin; J. I. Siepmann; "Calculating Gibbs free energies of transfer from Gibbs ensemble Monte Carlo simulations", *Theor. Chem. Acc.* **99** 347-350 (1998).
- M. G. Martin; J. I. Siepmann; "Novel configurational-bias Monte Carlo method for branched molecules. Transferable potentials for phase equilibria. 2. united-atom description of branched alkanes", *J. Phys. Chem. B* **103** 4508-4517 (1999).
- C. D. Wick; M. G. Martin; J. I. Siepmann; "Transferable potentials for phase equilibria. 4. United-atom description of linear and branched alkenes and alkylbenzenes", *J. Phys. Chem. B* **104** 8008-8016 (2000).
- B. Chen; J. Potoff; J. I. Siepmann; "Monte Carlo calculations for alcohols and their mixtures with alkanes. Transferable potentials for phase equilibria. 5. United-atom description of primary, secondary, and tertiary alcohols", *J. Phys. Chem. B* **105** 3090-3104 (2001).
- J. Potoff; J. I. Siepmann; "Vapor-Liquid Equilibria of Mixtures Containing Alkanes, Carbon Dioxide, and Nitrogen", *AIChE J.* **47** 1676-1682 (2001).
- G. Kamath; F. Cao; J. J. Potoff; "An Improved Force Field for the Prediction of the Vapor-Liquid Equilibria for Carboxylic Acids", *J. Phys. Chem. B* **108** 14130-14136 (2004).
- J. M. Stubbs; J. J. Potoff; J. I. Siepmann; "Transferable Potentials for Phase Equilibria. 6. United-Atom Description for Ethers, Glycols, Ketones, and Aldehydes", *J. Phys. Chem. B* **108** 17596-17605 (2004).

Force Fields: Universal Force Field (UFF)

- A. K. Rappe; W. A. Goddard III; "Charge Equilibration for Molecular Dynamics Simulations", *J. Phys. Chem.* **95** 3358-3363 (1991).
- A. K. Rappe; C. J. Casewit; K. S. Colwell; W. A. Goddard III; W. M. Skiff; "UFF, a Full Periodic Table Force Field for Molecular Mechanics and Molecular Dynamics Simulations", *J. Am. Chem. Soc.* **114** 10024-10035 (1992).

Force Fields: Vega *et al.* 1992 (Vega1992)

- L. Vega; E. de Miguel; L. F. Rull; G. Jackson; I. A. McLure; "Phase equilibria and critical behavior of square-well fluids of variable width by Gibbs ensemble Monte Carlo simulation", *J. Chem. Phys.* **96** 2296-2305 (1992).

Force Fields: Walther *et al.* 2001 (Walt2001)

- J. H. Walther; R. Jaffe; T. Halicioglu; P. Koumoutsakos; "Carbon Nanotubes in Water: Structural Characteristics and Energetics", *J. Phys. Chem. B* **105** 9980-9987 (2001).

Functional Forms: Surface Potentials

- W. A. Steele; "The Physical Interaction of Gases With Crystalline Solids", *Surf. Sci.* **36** 317-352 (1973).
- C. Lastoskie; K. E. Gubbins; N. Quirke; "Pore Size Heterogeneity and the Carbon Slit Pore: A Density Functional Theory Model", *Langmuir* **9** 2693-2702 (1993).
- J. B. Hooper; J. D. McCoy; J. G. Curro; "Density functional theory of simple polymers in a slit pore. I. Theory and efficient algorithm.", *J. Chem. Phys.* **112** 3090-3093 (2000).

Methods: Charge Assignment

- J. Gasteiger; M. Marsili; "Iterative Partial Equalization of Orbital Electronegativity - A Rapid Access To Atomic Charges", *Tetrahedron* **36** 3219-3288 (1980).

Methods: Configurational-bias

- J. I. Siepmann; "A method for the direct calculation of chemical potentials for dense chain systems", *Mol. Phys.* **70** 1145-1158 (1990).
- J. I. Siepmann; D. Frenkel; "Configurational bias Monte Carlo: a new sampling scheme for flexible chains", *Mol. Phys.* **75** 59-70 (1992).
- D. Frenkel; G. C. A. M. Mooij; B. Smit; "Novel scheme to study structural and thermal properties of continuously deformable molecules", *J. Phys.: Condens. Matter* **4** 3053-3076 (1992).
- M. Laso; J. J. de Pablo; U. W. Suter; "Simulation of phase equilibria for chain molecules", *J. Chem. Phys.* **97** 2817-2819 (1992).
- J. I. Siepmann; I. R. McDonald; "Monte Carlo simulations of mixed monolayers", *Mol. Phys.* **75** 255-259 (1992).
- T. J. H. Vlugt; M. G. Martin; B. Smit; J. I. Siepmann; R. Krishna; "Improving the efficiency of the configurational-bias Monte Carlo algorithm", *Mol. Phys.* **94** 727-733 (1998).
- T. J. H. Vlugt; R. Krishna; B. Smit; "Molecular Simulations of Adsorption Isotherms for Linear and Branched Alkanes and Their Mixtures in Silicalite" *J. Phys. Chem. B* **103** 1102-1118 (1999).
- M. G. Martin; J. I. Siepmann; "Novel configurational-bias Monte Carlo method for branched molecules. Transferable potentials for phase equilibria. 2. united-atom description of branched alkanes", *J. Phys. Chem. B* **103** 4508-4517 (1999).
- C. D. Wick; J. I. Siepmann; "Self-Adapting Fixed-End-Point Configurational-Bias Monte Carlo Method for the Regrowth of Interior Segments of Chain Molecules with Strong Intramolecular Interactions", *Macromolecules* **33** 7207-7218 (2000).
- M. G. Martin; A. P. Thompson; "Industrial property prediction using Towhee and LAMMPS" *Fluid Phase Equilib.* **217** 105-110 (2004).
- M. G. Martin; M. J. Biddy; "Monte Carlo Molecular Simulation Predictions for the Heat of Vaporization of Acetone and Butyramide", *Fluid Phase Equilib.* **236** 53-57 (2005).
- M. G. Martin; A. L. Frischknecht; "Using arbitrary trial distributions to improve intramolecular sampling in configurational-bias Monte Carlo", *Mol. Phys.* **104** 2439-2456 (2006).

Methods: Energy Biasing

- R. Q. Snurr; A. T. Bell; D. N. Theodorou; "Prediction of Adsorption of Aromatic Hydrocarbons in Silicalite from Grand Canonical Monte Carlo Simulations with Biased Insertions" *J. Phys. Chem.* **97** 13742 (1993).

Methods: Ensembles

- I. R. McDonald; "NpT-ensemble Monte Carlo calculations for binary liquid mixtures", *Mol. Phys.* **23** 41-58 (1972).
- S. Yashonath; C. N. R. Rao; "A Monte Carlo study of crystal structure transformations", *Mol. Phys.* **54** 245-251 (1985).
- A. Z. Panagiotopoulos; "Direct determination of phase coexistence properties of fluids by Monte Carlo simulation in a new ensemble", *Mol. Phys.* **61** 813-826 (1987).
- A. Z. Panagiotopoulos; N. Quirke; M. Stapleton; D. J. Tildesley; "Phase equilibria by simulation in the Gibbs ensemble. Alternative derivation, generalization and application to mixture and membrane equilibria", *Mol. Phys.* **63** 527-545 (1988).
- B. Smit; Ph. de Smedt; D. Frenkel; "Computer simulations in the Gibbs ensemble", *Mol. Phys.* **68** 931-950 (1989).

Methods: Ewald Sum and Electrostatics

- P. P. Ewald; *Ann. Phys.* **64** 253 (1921)
- M. J. L. Sangster; M. Dixon; "Interionic potentials in alkali halides and their use in simulations of the molten salts", *Advances in Physics* **25** 247-343 (1976).
- G. Hummer; L. R. Pratt; A. E. Garcia; B. J. Berne; S. W. Rick; "Electrostatic Potentials and Free Energies of Solvation of Polar and Charged Molecules", *J. Phys. Chem. B* **101** 3017-3020 (1997).

Methods: General Simulation

- M. P. Allen; D. J. Tildesley; "Computer Simulation of Liquids", Oxford Science Publications: New York (1987 - and many reprintings), ISBN 0 19 855645 4(Paperback).
- M. G. Martin; "Simulation of Phase Equlibria", Ph.D. Thesis, University of Minnesota (1999).

Methods: Histogram Reweighting

- A. M. Ferrenberg; R. H. Swendsen; "Optimized Monte Carlo Data Analysis", *Phys. Rev. Lett.* **63** 1195-1198 (1989).
- R. H. Swendsen; "Modern methods of analyzing Monte Carlo computer simulations", *Physica A* **194** 53-62 (1993).
- J. J. Potoff; A. Z. Panagiotopoulos; "Critical point and phase behavior of the pure fluid and a Lennard-Jones mixture". *J. Chem. Phys.* **109** 10914-10920 (1998).
- A. Z. Panagiotopoulos; "Monte Carlo methods for phase equilibria of fluids", *J. Phys.: Condens. Matter* **12** R25-R52 (2000).

Methods: Mixing Rules

- M. Waldman; A. T. Hagler; "New Combining Rules for Rare Gas van der Waals Parameters", *J. Comp. Chem.* **14** 1077-1084 (1993).
- T. C. Beutler; A. E. Mark; R. C. van Schaik; P. R. Gerber; W. F. van Gunsteren; "Avoiding singularities and numerical instabilities in free energy calculations based on molecular simulations", *Chem. Phys. Lett.* **222** 529-539 (1994).
- M. R. Shirts; J. W. Pitera; W. C. Swope; V. S. Pande; "Extremely precise free energy calculations of amino acid side chain analogs: Comparison of common molecular mechanics force fields for proteins", *J. Chem. Phys.* **119** 5740-5761 (2003).
- J. D.. Weeks; D. Chandler; H. C. Andersen; "Role of repulsive forces in determinig the equilibrium structure of simple liquids." *J. Chem. Phys*, **54**, 5237-5247 (1971)

Methods: Monte Carlo Moves (other than configurational-bias)

- N. Metropolis; A. W. Rosenbluth; M. N. Rosenbluth; A. H. Teller; E. Teller; "Equation of State Calculations by Fast Computing Machines" *J. Chem. Phys.* **21** 1087-1092 (1953).
- M. Lal; "Monte Carlo Computer Simulation of Chain Molecules. I", *Mol. Phys.* **17** 57-64 (1969).
- R. F. Cracknell; D. Nicholson; N. G. Parsonage; H. Evans; "Rotational insertion bias: a novel method for simulating dense phases of structured particles, with particular application to water", *Mol. Phys.* **71** 931-943 (1990).
- L. R. Dodd; T. D. Boone; D. N. Theodorou; "A concerted rotation algorithm for atomistic Monte Carlo simulation of polymer melts and glasses", *Mol. Phys.* **78** 961-996 (1993).
- B. Chen; J. I. Siepmann; "A Novel Monte Carlo Algorithm for Simulating Strongly Associating Fluids: Applications to Water, Hydrogen Fluoride, and Acetic Acid", *J. Phys. Chem. B* **104** 8725-8734 (2000).
- B. Chen; J. I. Siepmann; "Improving the Efficiency of the Aggregation-Volume-Bias Monte Carlo Algorithm", *J. Phys. Chem. B* **105** 11275-11282 (2001).

Methods: Pressure

- W. R. Smith; D. Henderson; Y. Tago; "Mean spherical approximation and optimized cluster theory for the square-well fluid", *J. Chem. Phys.* **67** 5308-5316 (1977).
- G. Hummer; N. Gronbech-Jensen; M. Neumann; "Pressure calculation in polar and charged systems using Ewald summation: Results for the extended simple point charge model of water" *J. Chem. Phys.* **109** 2791-2797 (1998).

Methods: Random Number Generators

- F. James "A review of pseudorandom number generators", *Comp. Phys. Comm.* **60** 329-344 (1990).
- F. James "RANLUX: A Fortran implementation of the high-quality pseudorandom number generator of Luscher", *Comp. Phys. Comm.* **79** 111-114 (1994).

Return to the main towhee web page

Send comments to: Marcus G. Martin

Last updated: January 05, 2007