Tinker9 User Manual

Zhi Wang & Jay W. Ponder

Mar 27, 2023

CONTENTS

CHAPTER

ONE

INTRODUCTION

CHAPTER

TWO

INSTALLATION

2.1 Prerequisites

Hardware

A relatively recent NVIDIA GPU is mandatory for the GPU code. The oldest NVIDIA GPU Tinker9 has been tested on is GeForce GTX 675MX (compute capability 3.0).

Operating Systems and Compilers

OS and Toolchain	Version
OS	Linux, WSL2, macOS <= 10.13
CMake	>= 3.18
Fortran	GNU or Intel
C++	[a]
CUDA/nvcc	[b]
OpenACC/NVHPC/PGI	[c]

- [a] Recent C++ compiler that supports C++11 syntax.
- [b] GPU code only. Version >= 9.0.
- [c] Optional for the GPU code. A recent NVIDIA HPC SDK is preferred.
- [d] We have successfully built Tinker9 on Windows WSL2 Ubuntu with CUDA 11.0 and NVHPC 20.9. Please check this link for more

details.

Using NVIDIA HPC SDK on Clusters

Prior to rebranding, the current NVIDIA HPC SDK was known as the PGI compiler suite. During Jan. 2020 we worked on a cluster that was still running Red Hat with gcc 4.8.5 by default without root privilege. Although several more recent gcc and PGI versions were available via the *module* program, the most recent PGI compiler (2019) was still configured with gcc 4.8.5 by default, which had a very bad support for C++11. Without root privilege on the cluster, we had to use a custom *localrc* file by running the following command to reconfigure PGI compiler with gcc 7.4.0:

```
makelocalrc $PGI/linux86-64-llvm/2019 \
-gcc /usr/local/gcc-7.4.0/bin/gcc \
-gpp /usr/local/gcc-7.4.0/bin/g++ \
-g77 /usr/local/gcc-7.4.0/bin/gfortran \
-d /dir_for_new_config -x
```

then added *export PGI_LOCALRC=/dir_for_new_config/localrc* to my bash resource file.

(Updated in Apr. 2021)

Compilers with the new brand name NVHPC now have a different directory structure. The new command will look like

```
makelocalrc $NVHPC/Linux_x86_64/2020/compilers \
-gcc /usr/local/gcc-7.4.0/bin/gcc \
-gpp /usr/local/gcc-7.4.0/bin/g++ \
-g77 /usr/local/gcc-7.4.0/bin/gfortran \
-d /dir_for_new_config -x
```

then add *export NVLOCALRC=/dir_for_new_config/localrc* to the shell resource file.

FFTW Libraries

Canonical Tinker requires FFTW libraries because by default it is compiled with OpenMP. Otherwise, Tinker will use *FFTPACK*. In Tinker9, the underlying *libtinker.a* will be compiled without OpenMP, therefore FFTW libraries are no longer mandatory for GPU code.

However, FFTW libraries are required by CPU code. Two prebuilt FFTW libraries, *libfftw3* and *libfftw3_threads* are needed by the double precision CPU code. The other two FFTW libraries, *libfftw3f* and *libfftw3f_threads* are needed by the mixed and single precision CPU code.

Other Nonmandatory Utilities

- ClangFormat: to format the source code.
- Doxygen: to generate developer guides.
- Sphinx: to generate user manual.
  - PDF version also depends on TeX.

```
python3 -m venv env-tinker9doc
source env-tinker9doc/bin/activate
pip3 install -r path_to/doc/manual/requirements.txt
```

2.2 Download the Canonical Tinker

Using the incorrect Tinker version, the executables are likely to fail with segfault. Since **d71f4793** (Dec. 6, 2021), downloading the required Tinker version is automated in the CMake script. For versions prior to this commit, please refer to the following.

Deprecated

If this source code was cloned by Git, you can checkout Tinker from the *tinker* Git submodule:

```
# checkout Tinker
cd tinker9
git submodule update --init
```

Alternatively, remove the directory *tinker9/tinker* and clone Tinker from GitHub to replace the deleted directory, then checkout the required version **b92eacc8**.

```
cd tinker9
rm -rf tinker
git clone https://github.com/tinkertools/tinker
cd tinker
git checkout <TheRequiredVersion>
```

2.3 Build Tinker9 with CMake

2.3.1 Quick Start

For a GPU card with compute capability 7.0, an example to compile the GPU code without OpenACC:

```
cd tinker9 && mkdir build
FC=gfortran compute_capability=70 gpu_lang=cuda cmake ..
make
make test
```

Assuming separate CUDA and NVHPC are properly installed, another example to compile the GPU code with both OpenACC and CUDA:

```
cd tinker9 && mkdir build
cmake -DCMAKE_Fortran_COMPILER=gfortran -DCOMPUTE_CAPABILITY=70 .
↪.
make
make test
```

For the options of other GPU devices and features, please refer to the subsequent sections.

2.3.2 Configure CMake

You can skip this section if you are familar with CMake.

Suppose the current working directory is */home/tinker9* and we want to create a build directory called *build* in */home/tinker9*. We can do *mkdir build* then *cd build*. Because the top-level CMakeLists.txt file is in the parent directory, if there was nothing else to configure, command *cmake ..* would generate the Makefile. The alternative way is to specify the build and source directories to CMake, e.g.,

```
cmake -B /home/tinker9/build -S /home/tinker9
```

Some CMake installations also provide a command line gui *ccmake* and a simple gui program *cmake-gui* that can replace *cmake* in the commands above.

2.3.3 Configure Compilers

If we are lucky, we do not need to specify compilers in the *cmake* configuration. However, specifying these compilers is preferred because programs are not always installed the way we wanted. Set *CXX=...*, *CUDACXX=...*, and *FC=...* to specify the non-default C++, CUDA, and Fortran compilers, respectively. These environmental variables are supported by *cmake*. Do not use nvfortran.

This cmake script checks a custom environmental variable *ACC=...* *only* for the OpenACC GPU code. If not set, the building script will take a guess at the OpenACC compiler. *ACC* will also be used as the C++ compiler. The value of *CXX* (if set) will be neglected.

The only place where a C compiler may be used in Tinker9 is on the old Mac computers that had Nvidia support. clang is hardwired in the cmake scripts to compile the Objective-C and C source files. Thus, *CC=...* is not worth setting in the cmake configuration.

2.3.4 Configure Tinker9

The following options are passed to CMake program with their default values (if there is one). **-D** is prefixed to the options. CMake provides two standard ways to let users customize the values:

- Change their values interactively in the *ccmake* command line gui;
- Pass the new value to CMake via command line arguments *cmake -DOPTION=NewValue*.

In addition to these two canonical methods, default value can also be set by its corresponding environmental variable, documented as **(env)** here. Note that there is no **-D** prefix for the environmental variables.

-DCMAKE_BUILD_TYPE (opt) = Release

Standard *CMAKE_BUILD_TYPE* option. Build type is case insensitive and can be *Debug*, *Release*, *RelWithDebInfo* (release with debug info), and *MinSizeRel* (minimum size release).

-DCMAKE_INSTALL_PREFIX (install) = [NO DEFAULT VALUE]

Install the executables under *${CMAKE_INSTALL_PREFIX}*. If this option is not set, *make install* is configured not to install anything, which is different from the default cmake behavior to install the program under */usr/local*.

-DSTD (std) = AUTO

C++ syntax standard. The source code is c++11-compliant, and should have no problems compiled with c++14. If set to *14* here, users should make sure the compilers are c++14-compliant. In general, users should not worry about the C++ standard for Tinker9. Using a more recent C++ standard to write the source code is unlikely to speed up the performance of Tinker9 and may harm the availability of Tinker9 to older machines.

-DPREC (prec) = mixed

Precision of the floating-point numbers. With flag *double/d*, all of the floating-point numbers are treated as real*8/double values, or real*4/single values if with flag *single/s*. Mixed precision flag *mixed/m* will use real*4 or real*8 numbers in different places. Note that this flag

will not change the precision of the variables hard-coded as *float* or *double* types.

-DDETERMINISTIC_FORCE (deterministic_force) = AUTO

Flag to use deterministic force. This feature will be implicitly enabled by mixed and single precisions, but can be explicitly disabled by setting the flag to *OFF* (or 0), and can be explicitly enabled by value *ON* (or 1).

In general, evaluating energy, forces etc. twice, we don't expect to get two identical answers, but we may not care as much because the difference is usually negligible. (See Why is cos(x) != cos(y)?) Whereas in MD, two simulations with the same initial configurations can easily diverge due to the accumulated difference. If, for whatever reason, you are willing to elongate the process of the inevitable divergence at the cost of slightly slower simulation speed, a more "deterministic" force (using fixed-point arithmetic) can help.

-DHOST (host) = OFF

Flag to compile to GPU (with value 0 or OFF) or CPU (with value 1 or ON) version.

-DGPU_LANG (gpu_lang) = OPENACC

If set to *CUDA*, the GPU code will only use the cuda source files. And the program will crash at runtime if it falls into an OpenACC code path.

-DCOMPUTE_CAPABILITY (compute_capability) = AUTO

GPU code only.

CUDA compute capability (multiplied by 10) of GPU. Valid values (non-inclusive) are 35, 50, 60, 70, 75, etc., and can be comma-separated, e.g. 35,60. Multiple compute capabilites will increase the size of executables. If left unspecified, the script will attempt to detect the GPU, although the detection may fail due to different reasons, which would then require this option to be specified explicitly.

If new cards are released but the newer compute capabilities are not supported, please inform us.

The full list of compute capabilities can be found on the NVIDIA website.

-DCUDA_DIR (cuda_dir) = /usr/local/cuda

Nvidia GPU code only.

Top-level CUDA installation directory, under which directories *include*, *lib* or *lib64* can be found. This option will supersede the CUDA installation identified by the official *CUDACXX* environmental variable.

Sometimes the PGI compiler and the NVCC compiler are not "compatible." For instance, although PGI 19.4 supports CUDA 9.2, 10.0, 10.1, but the default CUDA version configured in PGI 19.4 may be 9.2 and the external NVCC version is 10.1. One solution is to pass *CUDA_HOME=${cuda_dir}* to the PGI compiler, in which case, **cuda_dir** should be set to */usr/local/cuda-10.1*.

-DFFTW_DIR (fftw_dir) = ${CMAKE_BINARY_DIR}/fftw

CPU code only.

Top-level FFTW3 installation, under which *include/fftw3.h* and *lib/libfftw3* static libraries are expected to be found.

2.3.5 Make Tinker9

The following Makefile targets will be generated by CMake. Run *make -j* for the default target(s) and *make TARGET(S) -j* for others.

tinker9

Compile and link the *tinker9* executable.

all.tests

Compile and link the *all.tests* executable.

default

Make two targets: *tinker9* and *all.tests* executables.

all

Same as the default target.

test

Run unit tests in a random order. Exit on the first error.

man

Generate user manual.

doc

Generate developer guides.

CHAPTER

THREE

TUTORIALS

Let's create a directory *tutorial* under the home directory and copy the directories *tinker9/example* and *tinker9/params* here.

```
zw@Blade:~$ mkdir ~/tutorial; cd ~/tutorial
zw@Blade:~/tutorial$ cp -r ~/tinker9/{example,params} .
zw@Blade:~/tutorial$ ls
example/  params/
zw@Blade:~/tutorial$ cd example; ls
ar94.key  dhfr2.key  dhfr.key  dhfr.seq
ar94.xyz  dhfr2.xyz  dhfr.pdb  dhfr.xyz
```

This tutorial also assumes the executable *tinker9* is in your *PATH*. If not, prefix the directory to the *tinker9* executable.

3.1 Common File Types

You will find *.xyz* and *.key* files under *example* directory and *.prm* files under *params* directory.

sample.xyz The *.xyz* file is the basic Tinker Cartesian coordinates file type. It contains a title line followed by one line for each atom in the structure. Each line contains: the sequential number within the structure, an atomic symbol or name, X-, Y-, and Z-coordinates, the force field atom type number of the atom, and a list of the atoms

connected to the current atom. Except for programs whose basic operation is in torsional space, all Tinker calculations are done from some version of the *.xyz* format.

sample.key The keyword parameter file always has the extension *.key* and is optionally present during Tinker calculations. It contains values for any of a wide variety of switches and parameters that are used to change the course of the computation from the default. The detailed contents of this file is explained in a latter section. If a molecular system specific keyfile, in this case *sample.key*, is not present, the the Tinker program will look in the same directory for a generic file named *tinker.key*.

sample.prm The potential energy parameter files distributed with the Tinker package all end in the extension *.prm*, although this is not required by the programs themselves. Each of these files contains a definition of the potential energy functional forms for that force field as well as values for individual energy parameters. For example, the *mm3pro.prm* file contains the energy parameters and definitions needed for a protein-specific version of the MM3 force field.

3.2 Command Line GUI

Tinker programs can take interactive input. The prompt messages try to be self-explanatory. Following is an example to use the interactive interface for a short DHFR simulation with an *AMOEBA* force field.

Warning: The simulation will not run unless more keywords are added to the key file. This is only a demonstration to the interactive interface.

Item	Value	Input
Coordinate File	dhfr2.xyz	dhfr2.xyz
Simulations Steps	1000	1000
Time Step	2 fs	2
Time Between Saves	1 ps	1
Ensemble	NVT	2
Thermostat Temperature	298 K	298

```
zw@Blade:~/tutorial/example$ tinker9 dynamic

     ###############################################################
↪###
   ###############################################################
↪#####
  ###                                                            ␣
↪    ###
 ###         Tinker9  ---  Software Tools for Molecular Design    ␣
↪     ###
 ##                                                             ␣
↪      ##
 ##                     Version 1.0.0-rc   Jan 2021             ␣
↪      ##
 ###                        All Rights Reserved                 ␣
↪     ###
  ###                                                          ␣
↪    ###
   ###############################################################
↪#####
     ###############################################################
↪###
 Compiled at:  20:54:14  Feb  3 2021
 Commit Date:  Wed Feb 3 20:52:06 2021 -0600
 Commit:       3b0791c0


 Enter Cartesian Coordinate File Name :  dhfr2.xyz
```

(continues on next page)

(continued from previous page)

```
 ######################################################################
 Joint Amber-CHARMM Benchmark on Dihydrofolate Reductase in Water
 23558 Atoms, 62.23 Ang Cube, 9 Ang Nonbond Cutoffs, 64x64x64 PME
 ######################################################################

 Enter the Number of Dynamics Steps to be Taken :  1000

 Enter the Time Step Length in Femtoseconds [1.0] :  2

 Enter Time between Saves in Picoseconds [0.1] :  1

 Available Statistical Mechanical Ensembles :
    (1) Microcanonical (NVE)
    (2) Canonical (NVT)
    (3) Isoenthalpic-Isobaric (NPH)
    (4) Isothermal-Isobaric (NPT)
 Enter the Number of the Desired Choice  [1] :  2

 Enter the Desired Temperature in Degrees K [298] :  298
```

Once you are familar with the interactive interface, you can simply append the interactive input to the program name.

```
zw@Blade:~/tutorial/example$ tinker9 dynamic dhfr2.xyz 1000 2 1␣
↪2 298

     ##################################################################
↪###
   ###################################################################
↪#####
  ###                                                                 ␣
↪    ###
 ###         Tinker9  ---  Software Tools for Molecular Design       ␣
↪     ###
 ##                                                                  ␣
↪      ##
```

(continues on next page)

(continued from previous page)

```
 ##                    Version 1.0.0-rc   Jan 2021                 ␣
↪     ##
 ###                        All Rights Reserved                     ␣
↪    ###
  ###                                                               ␣
↪   ###
   ###################################################################
↪#####
    ###################################################################
↪###
 Compiled at:  20:54:14  Feb  3 2021
 Commit Date:  Wed Feb 3 20:52:06 2021 -0600
 Commit:       3b0791c0


 ###################################################################
 Joint Amber-CHARMM Benchmark on Dihydrofolate Reductase in Water
 23558 Atoms, 62.23 Ang Cube, 9 Ang Nonbond Cutoffs, 64x64x64 PME
 ###################################################################
```

You can also use *stdin redirection* to run Tinker programs. Save the command line arguments in a file and use it as follows.

```
zw@Blade:~/tutorial/example$ cat args.txt
dhfr2.xyz
1000
2
1
2
298
zw@Blade:~/tutorial/example$ tinker9 dynamic < args.txt
```

3.3 Program: analyze

3.4 Program: minimize

3.5 Program: dynamic

CHAPTER

FOUR

FEATURES & METHODS

4.1 Valence Potential Functions

4.1.1 Bond Stretching

Bond term is an empirical function of bond deviating from the ideal bond length, i.e., $\Delta b = b - b_0$. To support the AMOEBA force field model, Tinker includes the 3rd and 4th order terms.

$$U = k\Delta b^2(1 + k_3\Delta b + k_4\Delta b^2).$$

Setting 3rd and 4th order coefficients to zero will give the harmonic functional form.

Note: Different from Hooke's Law, $U = kx^2/2$, Tinker usually drops the coefficient $1/2$.

The Morse oscillator is also implemented in Tinker:

$$U = D_e[1 - \exp(-a\Delta b)]^2.$$

Parameter a is hardwired to 2 by approximation. Following equation $a = \sqrt{\frac{k}{2D_e}}$ and the Tinker convention to include $1/2$ in the force constant, De is $k/4$.

4.1.2 Angle Bending

Similar to bond stretching, angle bending term is also an empirical function of angle deviating from the ideal angle value, i.e., $\Delta\theta = \theta - \theta_0$. Terms from cubic to sextic are added to generalize the *HARMONIC* functional form.

$$U = k\Delta\theta^2(1 + k_3\Delta\theta + k_4\Delta\theta^2 + k_5\Delta\theta^3 + k_6\Delta\theta^4).$$

MMFF force field has a special treatment for *LINEAR* angle, e.g., carbon dioxide. Since the ideal angle should always be π rad, the deviation can be approximated by

$$\Delta\theta = \theta - \pi = 2(\frac{\theta}{2} - \frac{\pi}{2}) \sim 2\sin(\frac{\theta}{2} - \frac{\pi}{2}) = -2\cos\frac{\theta}{2}.$$

Only keeping the quadratic term, the angle bending term can be simplified to

$$U = 2k(1 + \cos\theta).$$

The *LINEAR* angle type is a special case of the SHAPES-style Fourier potential function [1] with 1-fold periodicity, which is referred to as the *FOURIER* angle type in Tinker jargon and has the following form

$$U = 2k(1 + \cos(n\theta - \theta_0)).$$

In addition, there is another *IN-PLANE* angle type for trigonal center atoms. One can project atom *D* to point *X* on plane *ABC*. Instead of angle *A-D-B*, the ideal and actual angle values are for angle *A-X-B* (see Fig. 4.1).

4.1.3 Stretch-Bend Coupling

The functional forms for bond stretching, angle bending, and stretch-bend coupling are those of the MM3 force field [2]:

$$U = (k_1\Delta b_1 + k_2\Delta b_2)\Delta\theta.$$

Even though force constants *k1* and *k2* are implemented as two independent variables in Tinker, they were treated as the same variable in the original literature.

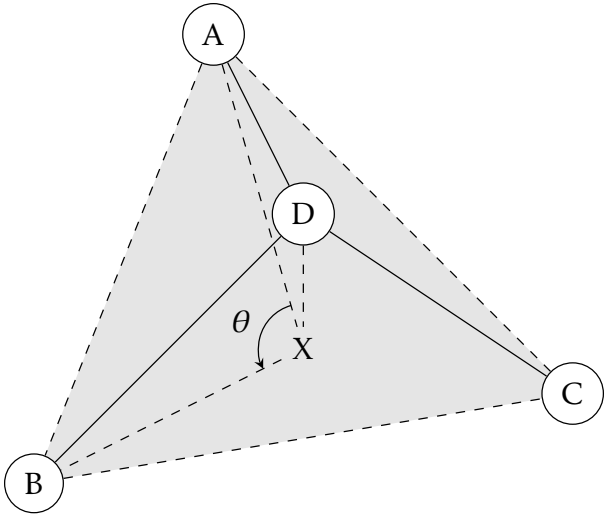

Fig. 4.1: A trigonal center and an in-plane angle.

4.1.4 Urey-Bradley Potential

Urey-Bradley potential energy accounts for the 1-3 nonbonded interactions. The cubic and quartic terms are added to the harmonic functional form in Tinker:

$$U = k\Delta u^2(1 + k_3\Delta u + k_4\Delta u^2),$$

where Δu is the difference of 1-3 distance deviating from its ideal value. Coefficients *k3* and *k4* are usually zero.

4.1.5 Out-of-Plane Bending

Tinker has implemented two types of out-of-plane bending angles. In the Wilson-Decius-Cross formulation [3], the out-of-plane angle χ associated with bond *BD* in Fig. 4.1 is the angle between *BD* and plane *ADC*, whereas the Allinger formulation uses the angle between *BD* and plane *ABC*. Similar to harmonic bond stretching, the following functional form has been

implemented in Tinker:

$$U = k\chi^2(1 + k_3\chi + k_4\chi^2).$$

4.1.6 Improper Dihedral

Commonly used in the CHARMM force fields, this potential function is meant to keep atoms planar. The ideal angle ϕ_0 defined by dihedral *D-A-B-C* will always be zero degrees. *D* is the trigonal atom, *A-B-C* are the peripheral atoms. In the original CHARMM parameter files, the trigonal atom is often listed last as *C-B-A-D*.

Some of the improper angles are "double counted" in the CHARMM protein parameter set. Since Tinker uses only one improper parameter per site, we have doubled these force constants in the Tinker version of the CHARMM parameters. Symmetric parameters, which are the origin of the "double counted" CHARMM values, are handled in the Tinker package by assigning all symmetric states and using the Tinker force constant divided by the symmetry number.

The harmonic functional form implemented in Tinker is

$$U = k(\phi - \phi_0)^2.$$

There is no coefficient $1/2$ before the force coefficient, which may be different in other software packages.

4.1.7 Improper Torsion

Commonly used in the AMBER force fields, it defines a hypothetical torsional angle for four atoms not successively bonded, e.g., a trigonal center. The functional form is similar to the proper torsion,

$$U = \sum_{n=1}^{3} V_n[1 + \cos(n\phi - \delta_n)],$$

where n is the periodicity and δ_n is the corresponding phase shift.

4.1.8 Torsional Angle

The functional form implemented in Tinker is

$$U = \sum_{n=1}^{6} V_n[1 + \cos(n\phi - \delta_n)],$$

where n is the periodicity (up to 6) and δ_n is the corresponding phase shift.

4.1.9 Pi-Orbital Torsional Angle

The 2-fold Fourier torsional angle potential to keep a pi-orbital molecular structure (e.g., ethylene) planar.

$$U = V[1 + \cos(2\phi - \pi)].$$

4.1.10 Stretch-Torsion Coupling

4.1.11 Angle-Torsion Coupling

4.1.12 Torsion-Torsion Coupling

The potential energy is extrapolated from a predefined 2-D map with the (ϕ, ψ) angles.

4.2 Van der Waals Potential Functions

4.3 Integrators and Ensembles

4.3.1 Monte Carlo Barostat

4.3.2 Berendsen Barostat

4.3.3 Verlet Integrator

4.3.4 RESPA Integrator

4.3.5 Extended Nosé-Hoover Chain

Authors of paper [4] (MTK) discussed several methods for NVT and NPT ensembles.

Number	Sections in MTK	Method
1a	2.1 4.3	NVT
2a	2.2 4.4	NPT (isotropic cell fluctuations)
3a	2.3 4.5	NPT (full cell fluctuations)
4a	5.2	XO-RESPA
4b	5.2	XI-RESPA
1b	5.3	RESPA 1a
2b	5.4	RESPA 2a
3b	5.4	RESPA 3a

The isothermal-isobaric integrator implemented in Fortran Tinker and here is NPT-XO (#2a-4a).

Tip: Nosé-Hoover Chain can be enabled by keywords

```
integrator nose-hoover
```

or

```
thermostat nose-hoover
barostat   nose-hoover
```

with the NPT option in the *dynamic* program.

4.3.6 Langevin Piston

The Langevin piston method for constant pressure [5] is integrated in the Leapfrog framework.

Tip: Langevin Piston can be enabled by keywords

```
integrator lpiston
```

or

```
thermostat lpiston
barostat   lpiston
```

with the NPT option in the *dynamic* program.

CHAPTER

FIVE

ELECTROSTATICS

5.1 Permanent Multipole

5.1.1 Definitions and Units

The electrostatic potential at r due to the charge distribution nearby is

$$\phi(r) = \frac{1}{4\pi\epsilon_0} \int ds \frac{\rho(s)}{|r-s|}, \; ds = \mathrm{d}x\mathrm{d}y\mathrm{d}z. \tag{5.1}$$

Tinker uses a variable **electric** (in **chgpot** module) to represent the the factor $1/(4\pi\epsilon_0)$. Its default magnitude is 332.063713, which is a constant defined by variable **coulomb** (in **units** module), and its units are kcal/mol Å/e^2. The default value can be modified by the *ELECTIRIC* keyword.

Note: Should the value of **coulomb** documented here be out-dated and become inconsistent with our code, please send us a pull request.

Expanding $1/|r-s|$ in Taylor series, $4\pi\epsilon_0\phi(r)$ can be rewritten as

$$\left[\int ds\rho(s)\right]\frac{1}{r} + \sum_i \left[\int ds\rho(s)s_i\right]\nabla_i\frac{1}{r} + \sum_{ij}\left[\frac{1}{2}\int ds\rho(s)s_is_j\right]\nabla_i\nabla_j\frac{1}{r} + \cdots, \tag{5.2}$$

where three pairs of square brackets give a set of definitions of monopole (charge, C), dipole (D), and quadrupole moments (Q^*), respectively. The

units of the multipole moments used in Tinker parameter files and internal calculation are different.

Multipole	Parameter Units	Internal Units
Charge	e	e
Dipole	e Bohr	e Å
Quadrupole	e Bohr^2	e Å^2

In addition to different units, the quadrupole moments in Tinker parameter files use what is traditionally called *traceless quadrupole* Θ that has a different definition than Q^*. The third term in (5.2) can be rewritten as

$$\sum_{ij}\left[\frac{1}{2}\int ds\rho(s)(3s_is_j - s^2\delta_{ij})\right]\frac{r_ir_j}{r^5},$$

hence the traceless quadrupole can be defined as

$$\Theta_{ij} = \frac{1}{2}\int ds\rho(s)(3s_is_j - s^2\delta_{ij}).$$

It is easy to confirm that $\sum_k^{x,y,z}(3s_ks_k - s^2) = 0$, thus,

$$\Theta_{ij} = 3Q^*_{ij} - \delta_{ij}\sum_k^{x,y,z} Q^*_{kk}.$$

Internally, Tinker scales Θ by $1/3$

$$Q = \Theta/3,$$

so that the energy expression is the same as if we were using Q^*.

5.1.2 Energy Torque Gradient

Potential energy

$$U = \frac{1}{4\pi\epsilon_0}\int ds\rho(s)\phi(s). \tag{5.3}$$

Potential energy with discretized charge distribution in (5.3)

$$U(r) = \phi(r)C(r) + \phi'(r)D(r) + \phi''(r)Q(r) + \cdots. \tag{5.4}$$

Distance

$$(r_x, r_y, r_z) = \boldsymbol{r} = r_2 - r_1.$$

Pairwise (atoms 1 and 2) quadrupole energy

$$U_{12} = M_1^T T_{12} M_2.$$

Multipoles

$$M_1 = \begin{pmatrix} C_1 \\ D_1 \\ Q_1 \end{pmatrix},\ M_2 = \begin{pmatrix} C_2 \\ D_2 \\ Q_2 \end{pmatrix}.$$

T matrix

$$T_{12} = \begin{pmatrix} 1 & \nabla_2 & \nabla_2^2 \\ \nabla_1 & \nabla_1\nabla_2 & \nabla_1\nabla_2^2 \\ \nabla_1^2 & \nabla_1^2\nabla_2 & \nabla_1^2\nabla_2^2 \end{pmatrix} \frac{1}{r}.$$

The upper left 4×4 elements of T_{12}

$$T_{12}^{4\times4} = \begin{pmatrix} 1/r & -r_x/r^3 & -r_y/r^3 & -r_z/r^3 \\ r_x/r^3 & -3r_x^2/r^5 + 1/r^3 & -3r_xr_y/r^5 & -3r_xr_z/r^5 \\ r_y/r^3 & -3r_xr_y/r^5 & -3r_y^2/r^5 + 1/r^3 & -3r_yr_z/r^5 \\ r_z/r^3 & -3r_xr_z/r^5 & -3r_yr_z/r^5 & -3r_z^2/r^5 + 1/r^3 \end{pmatrix}.$$

In the EWALD summation, $1/r^k$ terms will have different forms (Bn). Nevertheless, they are still connected through derivatives.

Non-EWALD	EWALD
$1/r$	$B_0 = \mathrm{erfc}(\alpha r)/r$
$1/r^3$	B_1
$3/r^5$	B_2
$15/r^7$	B_3
$105/r^9$	B_4
$945/r^{11}$	B_5

The *Bn* terms are related to the (complementary) Boys functions and (complementary) error functions. For $x > 0$ and $n \geq 0$,

$$\frac{\mathrm{erf}(x)}{x} = \frac{2}{\sqrt{\pi}} F_0(x^2),$$
$$\frac{\mathrm{erfc}(x)}{x} = \frac{2}{\sqrt{\pi}} F_0^C(x^2),$$
$$F_n(x) = \int_0^1 \exp(-xt^2) t^{2n} dt,$$
$$F_n^C(x) = \int_1^\infty \exp(-xt^2) t^{2n} dt.$$

The Boys functions can be generated through upward and downward recursions

$$F_n(x) = \frac{2xF_{n+1}(x) + \exp(-x)}{2n+1},$$
$$F_n^C(x) = \frac{2xF_{n+1}^C(x) - \exp(-x)}{2n+1}.$$

Energy, torque, and force

Terms	Energy	Torque	Force
C	ϕC	N/A	$\phi' C$
D	$\phi' D$	$\phi' \times D$	$\phi'' D$
Q	$\phi'' Q$	$\phi'' \times Q$	$\phi''' Q$

$$\tau(D) = \phi' \times D = D \times E,$$
$$\tau_i(Q) = -2 \sum_{jk} \sum_{l}^{xyz} \epsilon_{ijk} Q_{jl} \phi''_{kl},$$

where ϵ_{ijk} is the Levi-Civita symbol.

Reference: [6].

5.2 Induced Dipole

5.2.1 Energy

μ is the induced dipole in the external field E. The induced field due to the induced dipole is $E^u = -T\mu$, and the induced dipole is proportional to the total field E^t:

$$\mu = \alpha E^t = \alpha(E + E^u),$$

where α is the polarizability. Defining $\tilde{T} = \alpha^{-1} + T$, the induced dipole is the solution to the linear equation

$$\tilde{T}\mu = E. \tag{5.5}$$

The polarization energy is given by

$$\begin{aligned} U &= -\mu E + \int_0^\mu d\mu\, \tilde{T}\mu \\ &= -\mu E + \frac{1}{2}\mu\tilde{T}\mu. \end{aligned} \tag{5.6}$$

On the right-hand side of (5.6):

- the 1st term is the contribution from the external field;
- the 2nd term is the mutual and self polarization energy.

Finally, the polarization energy is

$$U = -\frac{1}{2}\mu E. \tag{5.7}$$

5.2.2 Gradient

With limited numerical precision, the solution to linear equation (5.5) cannot be fully precise:

$$\tilde{T}\mu = \epsilon + E. \tag{5.8}$$

The gradient of the induced dipole can be written in

$$\mu' = \tilde{T}^{-1}(\epsilon' + E' - \tilde{T}'\mu),$$

and the polarization gradient is

$$\begin{aligned} U' &= -\frac{1}{2}(E\mu' + \mu E') \\ &= -\frac{1}{2}[(-\epsilon + \mu\tilde{T})\tilde{T}^{-1}(\epsilon' + E' - \tilde{T}'\mu) + \mu E'] \\ &\approx -\frac{1}{2}(\mu E' - \mu\tilde{T}'\mu + \mu E'), \end{aligned}$$

only if the convergence of (5.8) is tight that ϵ and ϵ' terms will drop.

5.2.3 Conjugate Gradient

Tinker uses the following Conjugate Gradient algorithm (C.G.) with a sparse matrix preconditioner (denoted as M) [7] to obtain the induced dipoles. Related Tinker variables and routines are tabulated.

procedure CONJUGATE GRADIENT
  Guess Initial μ
  $r = E - \tilde{T}\mu$
  $p = Mr$
  while not converged **do**
    $\gamma \leftarrow rMr/p\tilde{T}p$
    $\mu \leftarrow \mu + \gamma p$
    $r_1 \leftarrow r - \gamma\tilde{T}p$
    $\beta \leftarrow r_1 M r_1 / rMr$ (Previous r is used.)
    $p \leftarrow Mr_1 + \beta p$
    Check Convergence
  end while
end procedure

C.G. Terms	Tinker variables and routines
γ	a
β	b
r	rsd
Mr	zrsd
p	conj
$\tilde{T}p$	vec
$-T$	ufield()
M	uscale()

5.2.4 Polarization Model: AMOEBA (Thole Damping 2)

AMOEBA force field adopts two polarization schemes, d and p, for the external field due to the permanent multipoles, and a third scheme u for mutual induced dipole interactions. Both d and u schemes are group-based. The p scheme is atomic connectivity-based. Tinker uses C.G. iterations to solve the following linear equations

$$(1/\alpha + T^u)\mu_d = E_d$$
$$(1/\alpha + T^u)\mu_p = E_p,$$

and defines the polarization energy as

$$U = -\frac{1}{2}\mu_d E_p. \tag{5.9}$$

From an optimizational perspective, (5.9) is the minimum of the target function

$$f_1(\mu_d, \mu_p) = \frac{1}{2}\left(\frac{1}{2}\mu_d \tilde{T}\mu_p + \frac{1}{2}\mu_p \tilde{T}\mu_d - E_d\mu_p - E_p\mu_d\right),$$

whereas the way C.G. coded in Tinker is to solve the minimum of another target function

$$f_2(\mu_d, \mu_p) = \frac{1}{2}\left(\frac{1}{2}\mu_d \tilde{T}\mu_d + \frac{1}{2}\mu_p \tilde{T}\mu_p - E_d\mu_d - E_p\mu_p\right).$$

The difference in two target functions is usually negligible unless other loose convergence methods are used to compute the induced dipoles.

In the Thole damping model, a charge distribution ρ is used as a replacement for the point dipole model. AMOEBA adopts the second functional form

$$\rho = \frac{3a}{4\pi} \exp(-au^3)$$

from paper [8], where u is the polarizability-scaled distance. The electrostatic field and potential at distance r can be obtained from Gauss's law,

$$E(r) = -\phi'(r) = \frac{1}{r^2} \int_0^u du\ 4\pi u^2 \rho = \frac{1 - \exp(-au^3)}{r^2},$$

$$\phi(r) = \int_r^\infty dr\ E(r) = \frac{\lambda_1}{r} = \frac{1}{r}\left[1 - \frac{(au^3)^{\frac{1}{3}}}{3}\Gamma(-\frac{1}{3}, au^3)\right],$$

where λ_1 serves as the B_0 term in EWALD quadrupole interactions. λ_n terms are also related via derivatives

$$\begin{aligned}
\phi'' &= \frac{1}{r^3}\left[2 - (2 + 3au^3)\exp(-au^3)\right], \\
\phi''' &= \frac{3}{r^4}\left[-2 + (2 + 2au^3 + 3a^2u^6)\exp(-au^3)\right], \\
\phi'''' &= \frac{3}{r^5}\left[8 - (8 + 8au^3 + 9a^3u^9)\exp(-au^3)\right],
\end{aligned}$$

$$\begin{aligned}
\phi'_i &= \phi' \frac{r_i}{r}, \\
\phi''_{ij} &= \left(\phi'' - \frac{\phi'}{r}\right)\frac{r_i r_j}{r^2} + \frac{\phi'}{r}\delta_{ij}, \\
\phi'''_{ijk} &= \left(\phi''' - \frac{3\phi''}{r} + \frac{3\phi'}{r^2}\right)\frac{r_i r_j r_k}{r^3} + \left(\frac{\phi''}{r} - \frac{\phi'}{r^2}\right)\frac{\sum r_k \delta_{ij}}{r}, \\
\phi''''_{ijkl} &= \left(\phi'''' - \frac{6\phi'''}{r} + \frac{15\phi''}{r^2} - \frac{15\phi'}{r^3}\right)\frac{r_i r_j r_k r_l}{r^4} \\
&\quad + \left(\frac{\phi'''}{r} - \frac{3\phi''}{r^2} + \frac{3\phi'}{r^3}\right)\frac{\sum r_k r_l \delta_{ij}}{r^2} + \left(\frac{\phi''}{r^2} - \frac{\phi'}{r^3}\right)\sum \delta_{kl}\delta_{ij}.
\end{aligned}$$

Thus,

$$
\begin{aligned}
-\lambda_3/r^3 &= \phi'/r \Rightarrow \\
\lambda_3 &= 1-\exp(-au^3), \\
3\lambda_5/r^5 &= (\phi''-\phi'/r)/r^2 \Rightarrow \\
\lambda_5 &= 1-(1+au^3)\exp(-au^3), \\
-15\lambda_7/r^7 &= (\phi'''-3\phi''/r+3\phi'/r^2)/r^3 \Rightarrow \\
\lambda_7 &= 1-\left(1+au^3+\frac{3}{5}a^2u^6\right)\exp(-au^3), \\
105\lambda_9/r^9 &= (\phi''''-6\phi'''/r+15\phi''/r^2-15\phi'/r^3)/r^4 \Rightarrow \\
\lambda_9 &= 1-\left(1+au^3+\frac{18}{35}a^2u^6+\frac{9}{35}a^3u^9\right)\exp(-au^3).
\end{aligned}
$$

5.3 Quasi-Internal Frame

5.3.1 Rotation Matrix

Consider two vectors u, v and two reference frames A, B. R is the rotation matrix of the *axes* such that

$$
\begin{aligned}
Ru_A &= u_B, \\
Rv_A &= v_B.
\end{aligned}
$$

Since $u_A^T v_A = u_B^T v_B$,

$$
R^T R = I.
$$

A 2-D tensor, e.g., quadrupole moment Q, in two reference frames are associated by

$$
u_A^T Q_A v_A = u_B^T Q_B v_B.
$$

It is easy to prove that

$$
RQ_A R^T = Q_B.
$$

Two common transformations used in Tinker are:

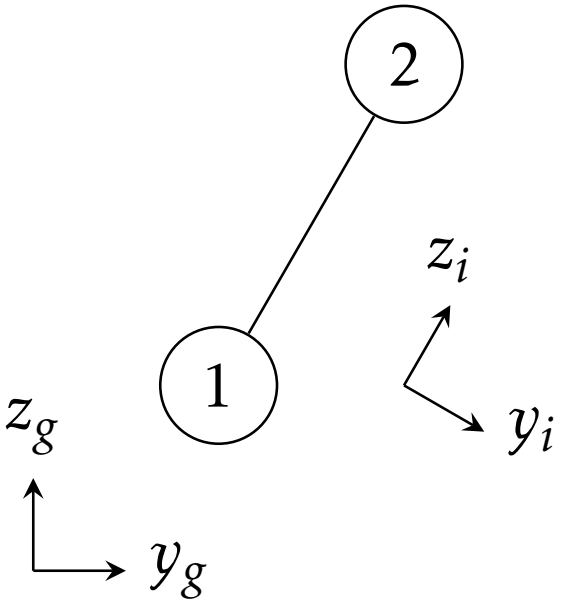

Fig. 5.1: Global frame *g* and QI frame *i* of atoms 1 and 2. The z direction of this QI frame is chosen along the distance vector.

- From (A) *Local Frame* (in which parameters are provided) to (B) *Global Frame* (in which the calculation is done);
- From (A) *Global Frame* (for direct pairwise electrostatics) to (B) *Quasi-Internal (QI) Frame* (for optimized algebra), as shown in Fig. 5.1.

5.3.2 Multipole Interaction in QI Frame

Once the distance vector is in QI frame, many derivatives can be simplified as shown in the following table. *f(r)* does not explicitly depend on r_x, r_y, r_z.

Gradients	Global Frame	QI Frame
$\partial f(r)/\partial x_2$	$f'(r)r_x/r$	0
$\partial f(r)/\partial y_2$	$f'(r)r_y/r$	0
$\partial f(r)/\partial z_2$	$f'(r)r_z/r$	$f'(r)$

For potential energy, (5.4) can be used without modification in QI frame. Since $\partial\phi_1/\partial z_1 = -E_{z1}$, the z direction gradient can be obtained from z direction electrostatic field (Ez):

$$\frac{\partial U}{\partial z} = -E_z C - E_z' D - E_z'' Q - \cdots.$$

Once the torques are computed the same way as in the previous section

$$\tau = \tau_1 + \tau_2 = \boldsymbol{r} \times \boldsymbol{F} = (U_x', U_y', U_z') \times (0, 0, r) = (U_y' r, -U_x' r, 0),$$

x and y direction gradients then become

$$U_x' = -\tau_y / r,$$
$$U_y' = \tau_x / r.$$

Depending on the direction of distance vector, the signs of x and y direction gradients may flip.

5.3.3 Details

In the following notes, $A : B$ stands for $A = A + B$. If there is no ambiguity, f' and f'' may stand for (f'_x, f'_y, f'_z) and $(f''_{xx}, f''_{yy}, f''_{zz}, f''_{xy}, f''_{xz}, f''_{yz})$, respectively.

Potential Terms	Notes
ϕ_1	ϕ_1
ϕ'_{1x}	$\partial\phi_1/\partial x_1$
ϕ'_{1y}	$\partial\phi_1/\partial y_1$
ϕ'_{1z}	$\partial\phi_1/\partial z_1$
ϕ''_{1xx}	$\partial^2\phi_1/\partial x_1^2$
ϕ''_{1yy}	$\partial^2\phi_1/\partial y_1^2$
ϕ''_{1zz}	$\partial^2\phi_1/\partial z_1^2$
ϕ''_{1xy}	$\partial^2\phi_1/\partial x_1\partial y_1$
ϕ''_{1xz}	$\partial^2\phi_1/\partial x_1\partial z_1$
ϕ''_{1yz}	$\partial^2\phi_1/\partial y_1\partial z_1$
ϕ_2	ϕ_2
ϕ'_{2x}	$\partial\phi_2/\partial x_2$
ϕ'_{2y}	$\partial\phi_2/\partial y_2$
ϕ'_{2z}	$\partial\phi_2/\partial z_2$
ϕ''_{2xx}	$\partial^2\phi_2/\partial x_2^2$
ϕ''_{2yy}	$\partial^2\phi_2/\partial y_2^2$
ϕ''_{2zz}	$\partial^2\phi_2/\partial z_2^2$
ϕ''_{2xy}	$\partial^2\phi_2/\partial x_2\partial y_2$
ϕ''_{2xz}	$\partial^2\phi_2/\partial x_2\partial z_2$
ϕ''_{2yz}	$\partial^2\phi_2/\partial y_2\partial z_2$

Charge Terms

$$\phi_1 : T_{12}^{(1,1)} C_2 = B_0 C_2,\ \phi_1' : T_{12}^{(2:4,1)} C_2 = \begin{pmatrix} 0 \\ 0 \\ r B_1 C_2 \end{pmatrix},$$

$$\phi_1'' : T_{12}^{(5:13,1)} C_2 = -\begin{pmatrix} B_1 C_2 \\ B_1 C_2 \\ (B_1 - r^2 B_2) C_2 \\ 0 \\ 0 \\ 0 \end{pmatrix}.$$

$$\phi_2 : T_{21}^{(1,1)} C_1 = B_0 C_1,\ \phi_2' : T_{21}^{(2:4,1)} C_1 = -\begin{pmatrix} 0 \\ 0 \\ r B_1 C_1 \end{pmatrix},$$

$$\phi_2'' : T_{21}^{(5:13,1)} C_1 = -\begin{pmatrix} B_1 C_1 \\ B_1 C_1 \\ (B_1 - r^2 B_2) C_1 \\ 0 \\ 0 \\ 0 \end{pmatrix}.$$

$$-E_{z1} : r B_1 C_2,\ -E_{z1}' : -\begin{pmatrix} 0 \\ 0 \\ B_1 - r^2 B_2 \end{pmatrix},$$

$$-E_{z1}'' : -\begin{pmatrix} r B_2 C_2 \\ r B_2 C_2 \\ (3 r B_2 - r^3 B_3) C_2 \\ 0 \\ 0 \\ 0 \end{pmatrix}.$$

Dipole Terms

$$\phi_1 : T_{12}^{(1,2:4)} D_2 = -rB_1 D_{z2},\ \phi_1' : T_{12}^{(2:4,2:4)} D_2 = \begin{pmatrix} B_1 D_{x2} \\ B_1 D_{y2} \\ (B_1 - r^2 B_2) D_{z2} \end{pmatrix},$$

$$\phi_1'' : T_{12}^{(5:13,2:4)} D_2 = \begin{pmatrix} rB_2 D_{z2} \\ rB_2 D_{z2} \\ (3rB_2 - r^3 B_3) D_{z2} \\ 0 \\ 2rB_2 D_{x2} \\ 2rB_2 D_{y2} \end{pmatrix}.$$

$$\phi_2 : T_{21}^{(1,2:4)} D_1 = rB_1 D_{z1},\ \phi_2' : T_{21}^{(2:4,2:4)} D_1 = \begin{pmatrix} B_1 D_{x1} \\ B_1 D_{y1} \\ (B_1 - r^2 B_2) D_{z1} \end{pmatrix},$$

$$\phi_2'' : T_{21}^{(5:13,2:4)} D_1 = -\begin{pmatrix} rB_2 D_{z1} \\ rB_2 D_{z1} \\ (3rB_2 - r^3 B_3) D_{z1} \\ 0 \\ 2rB_2 D_{x1} \\ 2rB_2 D_{y1} \end{pmatrix}.$$

$$-E_{z1} : (B_1 - r^2 B_2) D_{z2},\ -E_{z1}' : \begin{pmatrix} rB_2 D_{x2} \\ rB_2 D_{y2} \\ (3rB_2 - r^3 B_3) D_{z2} \end{pmatrix},$$

$$-E_{z1}'' : -\begin{pmatrix} (B_2 - r^2 B_3) D_{z2} \\ (B_2 - r^2 B_3) D_{z2} \\ (3B_2 - 6r^2 B_3 + r^4 B_4) D_{z2} \\ 0 \\ 2(B_2 - r^2 B_3) D_{x2} \\ 2(B_2 - r^2 B_3) D_{y2} \end{pmatrix}.$$

Quadrupole Terms

$$\phi_1 : T_{12}^{(1,5:13)} Q_2 = r^2 B_2 Q_{zz2},\ \phi_1' : T_{12}^{(2:4,5:13)} Q_2 = -\begin{pmatrix} 2rB_2 Q_{xz2} \\ 2rB_2 Q_{yz2} \\ (2rB_2 - r^3 B_3) Q_{zz2} \end{pmatrix},$$

$$\phi_1'' : T_{12}^{(5:13,5:13)} Q_2 = \begin{pmatrix} 2B_2 Q_{xx2} - r^2 B_3 Q_{zz2} \\ 2B_2 Q_{yy2} - r^2 B_3 Q_{zz2} \\ (2B_2 - 5r^2 B_3 + r^4 B_4) Q_{zz2} \\ 4B_2 Q_{xy2} \\ 4(B_2 - r^2 B_3) Q_{xz2} \\ 4(B_2 - r^2 B_3) Q_{yz2} \end{pmatrix}.$$

$$\phi_2 : T_{21}^{(1,5:13)} Q_1 = r^2 B_2 Q_{zz1},\ \phi_2' : T_{21}^{(2:4,5:13)} Q_1 = \begin{pmatrix} 2rB_2 Q_{xz1} \\ 2rB_2 Q_{yz1} \\ (2rB_2 - r^3 B_3) Q_{zz1} \end{pmatrix},$$

$$\phi_2'' : T_{21}^{(5:13,5:13)} Q_1 = \begin{pmatrix} 2B_2 Q_{xx1} - r^2 B_3 Q_{zz1} \\ 2B_2 Q_{yy1} - r^2 B_3 Q_{zz1} \\ (2B_2 - 5r^2 B_3 + r^4 B_4) Q_{zz1} \\ 4B_2 Q_{xy1} \\ 4(B_2 - r^2 B_3) Q_{xz1} \\ 4(B_2 - r^2 B_3) Q_{yz1} \end{pmatrix}.$$

$$-E_{z1} : -(2rB_2 - r^3 B_3) Q_{zz2},\ -E_{z1}' : \begin{pmatrix} 2(B_2 - r^2 B_3) Q_{xz2} \\ 2(B_2 - r^2 B_3) Q_{yz2} \\ (2B_2 - 5r^2 B_3 + r^4 B_4) Q_{zz2} \end{pmatrix},$$

$$-E_{z1}'' : \begin{pmatrix} -2rB_3 Q_{yy2} - r^3 B_4 Q_{zz2} \\ -2rB_3 Q_{xx2} - r^3 B_4 Q_{zz2} \\ (12rB_3 - 9r^3 B_4 + r^5 B_5) Q_{zz2} \\ 4rB_3 Q_{xy2} \\ 4(3rB_3 - r^3 B_4) Q_{xz2} \\ 4(3rB_3 - r^3 B_4) Q_{yz2} \end{pmatrix}.$$

CHAPTER

SIX

KEYWORDS

This section contains detailed descriptions of the keyword parameters used to define or alter the course of a Tinker calculation. The keyword control file is optional in the sense that all of the Tinker programs will run in the absence of a keyfile and will simply use default values or query the user for needed information. However, the keywords allow use of a wide variety of algorithmic and procedural options, many of which are unavailable interactively.

Keywords are read from the keyword control file. All programs look first for a keyfile with the same base name as the input molecular system and ending in the extension *key*. If this file does not exist, then Tinker tries to use a generic keyfile with the name *tinker.key* and located in the same directory as the input system. If neither a system-specific nor a generic keyfile is present, Tinker will continue by using default values for keyword options and asking interactive questions as necessary.

Tinker searches the keyfile during the course of a calculation for relevant keywords that may be present. All keywords must appear as the first word on the line. Any blank space to the left of the keyword is ignored, and all contents of the keyfiles are *case insensitive*. Some keywords take modifiers; i.e., Tinker looks further on the same line for additional information, such as the value of some parameter related to the keyword. Modifier information is read in free format, but must be completely contained on the same line as the original keyword. Any lines contained in the keyfile which do not qualify as valid keyword lines are treated as comments and are simply ignored.

Several keywords take a list of integer values (atom numbers, for example) as modifiers. For these keywords the integers can simply be listed explicitly and separated by spaces, commas or tabs. If a range of numbers is desired, it can be specified by listing the negative of the first number of the range, followed by a separator and the last number of the range. For example, the keyword line **LIGAND 4 -9 17 23** could be used to add atoms 4, 9 through 17, and 23 to the set of ligand atoms during a Tinker calculation.

Listed below are the available Tinker keywords sorted into groups by general function, along with brief descriptions of their actions, possible keyword modifiers, and usage examples.

6.1 Valence Potentials

6.1.1 Bond Stretching

BONDTYPE [HARMONIC / MORSE]

Chooses the functional form of the bond stretching potential. The *HARMONIC* option selects a Taylor series expansion containing terms from harmonic through quartic. The *MORSE* option selects a Morse potential fit to the ideal bond length and stretching force constant parameter values. The default is to use the *HARMONIC* potential.

BONDTERM [NONE / ONLY]

Controls use of the bond stretching potential energy term. In the absence of a modifying option, this keyword turns on use of the potential. The *NONE* option turns off use of this potential energy term. The *ONLY* option turns off all potential energy terms except for this one.

BONDUNIT [real]

Sets the scale factor needed to convert the energy value computed by the bond stretching potential into units of kcal/mol. The correct value is force field dependent and typically provided in the header of the master force field parameter file. The default value of 1.0 is used, if the *BONDUNIT* keyword is not given in the force field parameter file or the keyfile.

BOND [2 integers & 2 reals]

Provides the values for a single bond stretching parameter. The integer modifiers give the atom class numbers for the two kinds of atoms involved in the bond which is to be defined. The real number modifiers give the force constant value in kcal/mol/Å^2 for the bond and the ideal bond length in Angstroms. An example is as follows:

- BOND A B Force Ideal

BOND-CUBIC [real]

Sets the value (in 1/Å) of the cubic term in the Taylor series expansion form of the bond stretching potential energy. The real number modifier gives the value of the coefficient as a multiple of the quadratic coefficient. This term multiplied by the bond stretching energy unit conversion factor, the force constant, and the cube of the deviation of the bond length from its ideal value gives the cubic contribution to the bond stretching energy. The default value in the absence of the *BOND-CUBIC* keyword is zero; i.e., the cubic bond stretching term is omitted.

BOND-QUARTIC [real]

Sets the value (in 1/Å^2) of the quartic term in the Taylor series expansion form of the bond stretching potential energy. The real number modifier gives the value of the coefficient as a multiple of the quadratic coefficient. This term multiplied by the bond stretching energy unit conversion factor, the force constant, and the forth power of the deviation of the bond length from its ideal value gives the quartic contribution to the bond stretching energy. The default value in the absence of the *BOND-QUARTIC* keyword is zero; i.e., the quartic bond stretching term is omitted.

See also:

Bond Stretching

6.1.2 Angle Bending

ANGLETERM [NONE / ONLY]

Controls use of the bond angle bending potential energy term. In the absence of a modifying option, this keyword turns on use of the potential. The *NONE* option turns off use of this potential energy term. The *ONLY* option turns off all potential energy terms except for this one.

ANGLEUNIT [real]

Sets the scale factor needed to convert the energy value computed by the bond angle bending potential into units of kcal/mol. The correct value is force field dependent and typically provided in the header of the master force field parameter file. The default value of $(\pi/180)^2$ is used, if the *ANGLEUNIT* keyword is not given in the force field parameter file or the keyfile.

ANGLE [3 integers & 4 reals]

Provides the values for a single bond angle bending parameter. The integer modifiers give the atom class numbers for the three kinds of atoms involved in the angle which is to be defined. The real number modifiers give the force constant value for the angle and up to three ideal bond angles in degrees. In most cases only one ideal bond angle is given, and that value is used for all occurrences of the specified bond angle. If all three ideal angles are given, the values apply when the central atom of the angle is attached to 0, 1 or 2 additional hydrogen atoms, respectively. This "hydrogen environment" option is provided to implement the corresponding feature of Allinger's MM force fields. The default units for the force constant are kcal/mol/rad^2, but this can be controlled via the *ANGLEUNIT* keyword. An example is as follows:

- ANGLE A1 C A2 Force Ideal

ANGLEF [3 integers & 3 reals]

Provides the values for a single bond angle bending parameter for a SHAPES-style Fourier potential function. The integer modifiers give the atom class numbers for the three kinds of atoms involved in the angle which is to be defined. The real number modifiers give the force constant value for the angle, the angle shift in degrees, and the periodicity value.

Note that the force constant should be given as the "harmonic" value and not the native Fourier value. The default units for the force constant are kcal/mol/rad^2, but this can be controlled via the *ANGLEUNIT* keyword. An example is as follows:

- ANGLEF A1 C A2 Force Ideal Periodicity

ANGLEP [3 integers & 2 reals]

Provides the values for a single projected in-plane bond angle bending parameter. The integer modifiers give the atom class numbers for the three kinds of atoms involved in the angle which is to be defined. The real number modifiers give the force constant value for the angle and up to two ideal bond angles in degrees. In most cases only one ideal bond angle is given, and that value is used for all occurrences of the specified bond angle. If all two ideal angles are given, the values apply when the central atom of the angle is attached to 0 or 1 additional hydrogen atoms, respectively. This "hydrogen environment" option is provided to implement the corresponding feature of Allinger's MM force fields. The default units for the force constant are kcal/mol/rad^2, but this can be controlled via the *ANGLEUNIT* keyword. An example is as follows:

- ANGLEP A1 C A2 Force Ideal

ANGLE-CUBIC [real]

Sets the value (in 1/deg) of the cubic term in the Taylor series expansion form of the bond angle bending potential energy. The real number modifier gives the value of the coefficient as a multiple of the quadratic coefficient. This term multiplied by the angle bending energy unit conversion factor, the force constant, and the cube of the deviation of the bond angle from its ideal value gives the cubic contribution to the angle bending energy. The default value in the absence of the *ANGLE-CUBIC* keyword is zero; i.e., the cubic angle bending term is omitted.

ANGLE-QUARTIC [real]

Sets the value (in 1/deg^2) of the quartic term in the Taylor series expansion form of the bond angle bending potential energy. The real number modifier gives the value of the coefficient as a multiple of the quadratic coefficient. This term multiplied by the angle bending energy unit conversion factor, the force constant, and the forth power of the deviation of the bond angle from its ideal value gives the quartic contribution to the angle

bending energy. The default value in the absence of the *ANGLE-QUARTIC* keyword is zero; i.e., the quartic angle bending term is omitted.

ANGLE-PENTIC [real]

Sets the value (in $1/deg^3$) of the fifth power term in the Taylor series expansion form of the bond angle bending potential energy. The real number modifier gives the value of the coefficient as a multiple of the quadratic coefficient. This term multiplied by the angle bending energy unit conversion factor, the force constant, and the fifth power of the deviation of the bond angle from its ideal value gives the pentic contribution to the angle bending energy. The default value in the absence of the *ANGLE-PENTIC* keyword is zero; i.e., the pentic angle bending term is omitted.

ANGLE-SEXTIC [real]

Sets the value (in $1/deg^4$) of the sixth power term in the Taylor series expansion form of the bond angle bending potential energy. The real number modifier gives the value of the coefficient as a multiple of the quadratic coefficient. This term multiplied by the angle bending energy unit conversion factor, the force constant, and the sixth power of the deviation of the bond angle from its ideal value gives the sextic contribution to the angle bending energy. The default value in the absence of the *ANGLE-SEXTIC* keyword is zero; i.e., the sextic angle bending term is omitted.

See also:

Angle Bending

6.1.3 Stretch-Bend Coupling

STRBNDTERM [NONE / ONLY]

Controls use of the bond stretching-angle bending cross term potential energy. In the absence of a modifying option, this keyword turns on use of the potential. The *NONE* option turns off use of this potential energy term. The *ONLY* option turns off all potential energy terms except for this one.

STRBNDUNIT [real]

Sets the scale factor needed to convert the energy value computed by the bond stretching-angle bending cross term potential into units of kcal/mol. The correct value is force field dependent and typically provided in the header of the master force field parameter file. The default value of $\pi/180$ is used, if the *STRBNDUNIT* keyword is not given in the force field parameter file or the keyfile.

STRBND [3 integers & 2 reals]

Provides the values for a single stretch-bend cross term potential parameter. The integer modifiers give the atom class numbers for the three kinds of atoms involved in the angle which is to be defined. The real number modifiers give the force constant values for the first bond (first two atom classes) with the angle, and the second bond with the angle, respectively. The default units for the stretch-bend force constant are kcal/mol/Å/rad, but this can be controlled via the *STRBNDUNIT* keyword. An example is as follows:

- STRBND A1 C A2 Force1 Force2

See also:

Stretch-Bend Coupling

6.1.4 Urey-Bradley Potential

UREYTERM [NONE / ONLY]

Controls use of the Urey-Bradley potential energy term. In the absence of a modifying option, this keyword turns on use of the potential. The *NONE* option turns off use of this potential energy term. The *ONLY* option turns off all potential energy terms except for this one.

UREYUNIT [real]

Sets the scale factor needed to convert the energy value computed by the Urey-Bradley potential into units of kcal/mol. The correct value is force field dependent and typically provided in the header of the master force field parameter file. The default value of 1.0 is used, if the *UREYUNIT* keyword is not given in the force field parameter file or the keyfile.

UREYBRAD [3 integers & 2 reals]

Provides the values for a single Urey-Bradley cross term potential parameter. The integer modifiers give the atom class numbers for the three kinds of atoms involved in the angle for which a Urey-Bradley term is to be defined. The real number modifiers give the force constant value for the term and the target value for the 1-3 distance in Angstroms. The default units for the force constant are kcal/mol/Å^2, but this can be controlled via the *UREYUNIT* keyword. An example is as follows:

- UREYBRAD A1 C A3 Force Ideal

UREY-CUBIC [real]

Sets the value (in 1/Å) of the cubic term in the Taylor series expansion form of the Urey-Bradley potential energy. The real number modifier gives the value of the coefficient as a multiple of the quadratic coefficient. The default value in the absence of the *UREY-CUBIC* keyword is zero; i.e., the cubic Urey-Bradley term is omitted.

UREY-QUARTIC [real]

Sets the value (in 1/Å^2) of the quartic term in the Taylor series expansion form of the Urey-Bradley potential energy. The real number modifier gives the value of the coefficient as a multiple of the quadratic coefficient. The default value in the absence of the *UREY-QUARTIC* keyword is zero; i.e., the quartic Urey-Bradley term is omitted.

See also:

Urey-Bradley Potential

6.1.5 Out-of-Plane Bending

OPBENDTYPE [W-D-C / ALLINGER]

Sets the type of angle to be used in the out-of-plane bending potential energy term. The choices are to use the Wilson-Decius-Cross (W-D-C) formulation from vibrational spectroscopy, or the Allinger angle from the MM2/MM3 force fields. The default value in the absence of the *OPBENDTYPE* keyword is to use the W-D-C angle.

OPBENDTERM [NONE / ONLY]

Controls use of the out-of-plane bending potential energy term. In the absence of a modifying option, this keyword turns on use of the potential. The *NONE* option turns off use of this potential energy term. The *ONLY* option turns off all potential energy terms except for this one.

OPBENDUNIT [real]

Sets the scale factor needed to convert the energy value computed by the out-of-plane bending potential into units of kcal/mol. The correct value is force field dependent and typically provided in the header of the master force field parameter file. The default of $(\pi/180)^2$ is used, if the *OPBENDUNIT* keyword is not given in the force field parameter file or the keyfile.

OPBEND [4 integers & 1 real]

Provides the values for a single out-of-plane bending potential parameter. The first integer modifier is the atom class of the out-of-plane atom and the second integer is the atom class of the central trigonal atom. The third and fourth integers give the atom classes of the two remaining atoms attached to the trigonal atom. Values of zero for the third and fourth integers are treated as wildcards, and can represent any atom type. The real number modifier gives the force constant value for the out-of-plane angle. The default units for the force constant are kcal/mol/rad^2, but this can be controlled via the *OPBENDUNIT* keyword. An example is as follows:

- OPBEND A B 0 0 force

OPBEND-CUBIC [real]

Sets the value (in 1/deg) of the cubic term in the Taylor series expansion form of the out-of-plane bending potential energy. The real number modifier gives the value of the coefficient as a multiple of the quadratic coefficient. This term multiplied by the out-of-plane bending energy unit conversion factor, the force constant, and the cube of the deviation of the out-of-plane angle from zero gives the cubic contribution to the out-of-plane bending energy. The default value in the absence of the *OPBEND-CUBIC* keyword is zero; i.e., the cubic out-of-plane bending term is omitted.

OPBEND-QUARTIC [real]

Sets the value (in 1/deg^2) of the quartic term in the Taylor series expansion form of the out-of-plane bending potential energy. The real number modifier gives the value of the coefficient as a multiple of the quadratic

coefficient. This term multiplied by the out-of-plane bending energy unit conversion factor, the force constant, and the forth power of the deviation of the out-of-plane angle from zero gives the quartic contribution to the out-of-plane bending energy. The default value in the absence of the *OPBEND-QUARTIC* keyword is zero; i.e., the quartic out-of-plane bending term is omitted.

OPBEND-PENTIC [real]

Sets the value (in $1/\text{deg}^3$) of the fifth power term in the Taylor series expansion form of the out-of-plane bending potential energy. The real number modifier gives the value of the coefficient as a multiple of the quadratic coefficient. This term multiplied by the out-of-plane bending energy unit conversion factor, the force constant, and the fifth power of the deviation of the out-of-plane angle from zero gives the pentic contribution to the out-of-plane bending energy. The default value in the absence of the *OPBEND-PENTIC* keyword is zero; i.e., the pentic out-of-plane bending term is omitted.

OPBEND-SEXTIC [real]

Sets the value (in $1/\text{deg}^4$) of the sixth power term in the Taylor series expansion form of the out-of-plane bending potential energy. The real number modifier gives the value of the coefficient as a multiple of the quadratic coefficient. This term multiplied by the out-of-plane bending energy unit conversion factor, the force constant, and the sixth power of the deviation of the out-of-plane angle from zero gives the sextic contribution to the out-of-plane bending energy. The default value in the absence of the *OPBEND-SEXTIC* keyword is zero; i.e., the sextic out-of-plane bending term is omitted.

See also:

Out-of-Plane Bending

6.1.6 Improper Dihedral

IMPROPTERM [NONE / ONLY]

Controls use of the CHARMM-style improper dihedral angle potential energy term. In the absence of a modifying option, this keyword turns on use of the potential. The *NONE* option turns off use of this potential energy term. The *ONLY* option turns off all potential energy terms except for this one.

IMPROPUNIT [real]

Sets the scale factor needed to convert the energy value computed by the CHARMM-style improper dihedral angle potential into units of kcal/mol. The correct value is force field dependent and typically provided in the header of the master force field parameter file. The default value of $(\pi/180)^2$ is used, if the *IMPROPUNIT* keyword is not given in the force field parameter file or the keyfile.

IMPROPER [4 integers & 2 reals]

Provides the values for a single CHARMM-style improper dihedral angle parameter. The integer modifiers give the atom class numbers for the four kinds of atoms involved in the torsion which is to be defined. The real number modifiers give the force constant value for the deviation from the target improper torsional angle, and the target value for the torsional angle, respectively. The default units for the improper force constant are kcal/mol/rad^2, but this can be controlled via the *IMPROPUNIT* keyword.

The real number modifiers give the force constant in kcal/mol/rad^2 and ideal dihedral angle in degrees. An example is as follows:

- IMPROPER D A B C Force Ideal

See also:

Improper Dihedral

6.1.7 Improper Torsion

IMPTORTERM [NONE / ONLY]

Controls use of the AMBER-style improper torsional angle potential energy term. In the absence of a modifying option, this keyword turns on use of the potential. The *NONE* option turns off use of this potential energy term. The *ONLY* option turns off all potential energy terms except for this one.

IMPTORUNIT [real]

Sets the scale factor needed to convert the energy value computed by the AMBER-style improper torsional angle potential into units of kcal/mol. The correct value is force field dependent and typically provided in the header of the master force field parameter file. The default value of 1.0 is used, if the *IMPTORSUNIT* keyword is not given in the force field parameter file or the keyfile.

IMPTORS [4 integers & up to 3 real/real/integer triples]

Provides the values for a single AMBER-style improper torsional angle parameter. The first four integer modifiers give the atom class numbers for the atoms involved in the improper torsional angle to be defined. By convention, the third atom class of the four is the trigonal atom on which the improper torsion is centered. The torsional angle computed is literally that defined by the four atom classes in the order specified by the keyword. Each of the remaining triples of real/real/integer modifiers give the half-amplitude in kcal/mol, phase offset in degrees and periodicity of a particular improper torsional term, respectively. Periodicities through 3-fold are allowed for improper torsional parameters. An example is as follows:

- IMPTORS A B C D Amplitude PhaseOffset Periodicity

See also:

Improper Torsion

6.1.8 Torsional Angle

TORSIONTERM [NONE / ONLY]

Controls use of the torsional angle potential energy term. In the absence of a modifying option, this keyword turns on use of the potential. The *NONE* option turns off use of this potential energy term. The *ONLY* option turns off all potential energy terms except for this one.

TORSIONUNIT [real]

Sets the scale factor needed to convert the energy value computed by the torsional angle potential into units of kcal/mol. The correct value is force field dependent and typically provided in the header of the master force field parameter file. The default value of 1.0 is used, if the *TORSIONUNIT* keyword is not given in the force field parameter file or the keyfile.

TORSION [4 integers & up to 6 real/real/integer triples]

Provides the values for a single torsional angle parameter. The first four integer modifiers give the atom class numbers for the atoms involved in the torsional angle to be defined. Each of the remaining triples of real/real/integer modifiers give the amplitude in kcal/mol, phase offset in degrees and periodicity of a particular torsional function term, respectively. Periodicities through 6-fold are allowed for torsional parameters. An example is as follows:

- TORSION A B C D Amplitude PhaseOffset Periodicity

See also:

Torsional Angle

6.1.9 Pi-Orbital Torsional Angle

PITORSTERM [NONE / ONLY]

Controls use of the pi-orbital torsional angle potential energy term. In the absence of a modifying option, this keyword turns on use of the potential. The *NONE* option turns off use of this potential energy term. The *ONLY* option turns off all potential energy terms except for this one.

PITORSUNIT [real]

Sets the scale factor needed to convert the energy value computed by the pi-orbital torsional angle potential into units of kcal/mol. The correct value is force field dependent and typically provided in the header of the master force field parameter file. The default value of 1.0 is used, if the *PITORSUNIT* keyword is not given in the force field parameter file or the keyfile.

PITORS [2 integers & 1 real]

Provides the values for a single pi-orbital torsional angle potential parameter. The two integer modifiers give the atom class numbers for the atoms involved in the central bond of the torsional angle to be parameterized. The real modifier gives the value of the 2-fold Fourier amplitude in kcal/mol for the torsional angle between p-orbitals centered on the defined bond atom classes. The default units for the stretch-torsion force constant can be controlled via the *PITORSUNIT* keyword. An example is as follows:

- PITORS A B Amplitude

See also:

Pi-Orbital Torsional Angle

6.1.10 Stretch-Torsion Coupling

Stretch-Torsion Coupling

STRTORTERM [NONE / ONLY]

Controls use of the bond stretching-torsional angle cross term potential energy. In the absence of a modifying option, this keyword turns on use of the potential. The *NONE* option turns off use of this potential energy term. The *ONLY* option turns off all potential energy terms except for this one.

STRTORUNIT [real]

Sets the scale factor needed to convert the energy value computed by the bond stretching-torsional angle cross term potential into units of kcal/mol. The correct value is force field dependent and typically provided in the header of the master force field parameter file. The default

value of 1.0 is used, if the *STRTORUNIT* keyword is not given in the force field parameter file or the keyfile.

STRTORS

Provides the values for a single stretch-torsion cross term potential parameter. The two integer modifiers give the atom class numbers for the atoms involved in the central bond of the torsional angles to be parameterized. The real modifier gives the value of the stretch-torsion force constant for all torsional angles with the defined atom classes for the central bond. The default units for the stretch-torsion force constant can be controlled via the *STRTORUNIT* keyword.

6.1.11 Angle-Torsion Coupling

ANGTORTERM [NONE / ONLY]

Controls use of the angle bending-torsional angle cross term. In the absence of a modifying option, this keyword turns on use of the potential. The *NONE* option turns off use of this potential energy term. The *ONLY* option turns off all potential energy terms except for this one.

ANGTORUNIT [real]

Sets the scale factor needed to convert the energy value computed by the angle bending-torsional angle cross term into units of kcal/mol. The correct value is force field dependent and typically provided in the header of the master force field parameter file. The default value of $\pi/180$ is used, if the *ANGTORUNIT* keyword is not given in the force field parameter file or the keyfile.

ANGTORS [4 integers & 6 reals]

Provides the values for a single bond angle bending-torsional angle parameter. The integer modifiers give the atom class numbers for the four kinds of atoms involved in the torsion and its contained angles. The real number modifiers give the force constant values for both angles coupled with 1-, 2- and 3-fold torsional terms. The default units for the force constants are kcal/mol/rad, but this can be controlled via the *ANGTORUNIT* keyword.

See also:

Angle-Torsion Coupling

6.1.12 Torsion-Torsion Coupling

TORTORTERM [NONE / ONLY]

Controls use of the torsion-torsion potential energy term. In the absence of a modifying option, this keyword turns on use of the potential. The *NONE* option turns off use of this potential energy term. The *ONLY* option turns off all potential energy terms except for this one.

TORTORUNIT [real]

Sets the scale factor needed to convert the energy value computed by the torsion-torsion potential into units of kcal/mol. The correct value is force field dependent and typically provided in the header of the master force field parameter file. The default value of 1.0 is used, if the *TORTORUNIT* keyword is not given in the force field parameter file or the keyfile.

TORTORS [7 integers, then multiple lines of 2 integers and 1 real]

Provides the values for a single torsion-torsion parameter. The first five integer modifiers give the atom class numbers for the atoms involved in the two adjacent torsional angles to be defined. The last two integer modifiers contain the number of data grid points that lie along each axis of the torsion-torsion map. For example, this value will be 13 for a 30 degree torsional angle spacing, i.e., 360/30 = 12, but 13 values are required since data values for -180 and +180 degrees must both be supplied. The subsequent lines contain the torsion-torsion map data as the integer values in degrees of each torsional angle and the target energy value in kcal/mol.

See also:

Torsion-Torsion Coupling

6.2 Constraints & Restraints

RESTRAINTERM [NONE / ONLY]

Controls use of the restraint potential energy terms. In the absence of a modifying option, this keyword turns on use of these potentials. The *NONE* option turns off use of these potential energy terms. The *ONLY* option turns off all potential energy terms except for these terms.

RESTRAIN-ANGLE [3 integers & 3 reals]

Implements a flat-welled harmonic potential that can be used to restrain the angle between three atoms to lie within a specified angle range. The integer modifiers contain the atom numbers of the three atoms whose angle is to be restrained. The first real modifier is the force constant in kcal/mol/deg^2 for the restraint. The last two real modifiers give the lower and upper bounds in degrees on the allowed angle values. If the angle lies between the lower and upper bounds, the restraint potential is zero. Outside the bounds, the harmonic restraint is applied. If the angle range modifiers are omitted, then the atoms are restrained to the angle found in the input structure. If the force constant is also omitted, a default value of 10.0 is used.

RESTRAIN-DISTANCE [2 integers & 3 reals]

Implements a flat-welled harmonic potential that can be used to restrain two atoms to lie within a specified distance range. The integer modifiers contain the atom numbers of the two atoms to be restrained. The first real modifier specifies the force constant in kcal/mol/Å^2 for the restraint. The next two real modifiers give the lower and upper bounds in Angstroms on the allowed distance range. If the interatomic distance lies between these lower and upper bounds, the restraint potential is zero. Outside the bounds, the harmonic restraint is applied. If the distance range modifiers are omitted, then the atoms are restrained to the interatomic distance found in the input structure. If the force constant is also omitted, a default value of 100.0 is used.

RESTRAIN-GROUPS [2 integers & 3 reals]

Implements a flat-welled harmonic distance restraint between the centers-of-mass of two groups of atoms. The integer modifiers are the numbers of

the two groups which must be defined separately via the *GROUP* keyword. The first real modifier is the force constant in kcal/mol/Å^2 for the restraint. The last two real modifiers give the lower and upper bounds in Angstroms on the allowed intergroup center-of-mass distance values. If the distance range modifiers are omitted, then the groups are restrained to the distance found in the input structure. If the force constant is also omitted, a default value of 100.0 is used.

RESTRAIN-PLANE [X/Y/Z 1 integer & 3 reals]

Provides the ability to restrain an individual atom to a specified plane orthogonal to a coordinate axis. The first modifier gives the axis (X, Y or Z) perpendicular to the restraint plane. The integer modifier contains the atom number of the atom to be restrained. The first real number modifier gives the coordinate value to which the atom is restrained along the specified axis. The second real modifier sets the force constant in kcal/mol/Å^2 for the harmonic restraint potential. The final real modifier defines a range above and below the specified plane within which the restraint value is zero. If the force constant is omitted, a default value of 100.0 is used. If the exclusion range is omitted, it is taken to be zero.

RESTRAIN-POSITION [1 integer & 5 reals, OR 2 integers & 2 reals]

Provides the ability to restrain an atom or group of atoms to specified coordinate positions. An initial positive integer modifier contains the atom number of the atom to be restrained. The first three real number modifiers give the X-, Y- and Z-coordinates to which the atom is tethered. The fourth real modifier sets the force constant in kcal/mol/Å^2 for the harmonic restraint potential. The final real modifier defines a sphere around the specified coordinates within which the restraint value is zero. If the coordinates are omitted, then the atom is restrained to the origin. If the force constant is omitted, a default value of 100.0 is used. If the exclusion sphere radius is omitted, it is taken to be zero.

Alternatively, if the initial integer modifier is negative, then a second integer is read, followed by two real number modifiers. All atoms in the range from the absolute value of the first integer through the second integer are restrained to their current coordinates. The first real modifier is the harmonic force constant in kcal/mol/Å^2, and the second real defines a sphere around each atom within which the restraint value is zero. If the force constant is omitted, a default value of 100.0 is used. If the exclusion

sphere radius is omitted, it is taken to be zero.

RESTRAIN-TORSION [4 integers & 3 reals]

Implements a flat-welled harmonic potential that can be used to restrain the torsional angle between four atoms to lie within a specified angle range. The initial integer modifiers contains the atom numbers of the four atoms whose torsional angle, computed in the atom order listed, is to be restrained. The first real modifier gives a force constant in kcal/mol/deg^2 for the restraint. The last two real modifiers give the lower and upper bounds in degrees on the allowed torsional angle values. The angle values given can wrap around across -180 and +180 degrees. Outside the allowed angle range, the harmonic restraint is applied. If the angle range modifiers are omitted, then the atoms are restrained to the torsional angle found in the input structure. If the force constant is also omitted, a default value of 1.0 is used.

6.3 HIPPO Force Field

CHGTRN [integer & 2 reals]

DISPERSION [integer & 2 reals]

6.4 Molecular Dynamics and Ensembles

INTEGRATOR [VERLET / RESPA / NOSE-HOOVER / LPISTON]

See also:

Verlet Integrator, *RESPA Integrator*, *Extended Nosé-Hoover Chain*, *Langevin Piston*

THERMOSTAT [NOSE-HOOVER / LPISTON]

See also:

Extended Nosé-Hoover Chain, *Langevin Piston*

BAROSTAT [MONTECARLO / BERENDSEN / NOSE-HOOVER / LPISTON]

See also:

Monte Carlo Barostat, Berendsen Barostat, Extended Nosé-Hoover Chain, Langevin Piston

6.5 Free Energy

OSRW-ELE [ ]

OSRW-LAMBDA [real] Sets the internal logical flag for OSRW to *true* and provides the initial value of lambda.

OSRW-TORS [ ]

OSRW-VDW [ ]

ROTATABLE-BOND [integer list]

6.6 Parallelization

CUDA-DEVICE [integer]

Followed by an integer value starting from 0, sets the CUDA-enabled GPU device for the program. Value will be overwritten by environment variable *CUDA_DEVICE*. For instance, a node has four CUDA devices, and the *CUDA_VISIBLE_DEVICES* environment variable (part of CUDA library) has been set to *CUDA_VISIBLE_DEVICES=1,3*. This means only two CUDA devices are avaiable here, thus the valid values for *CUDA-DEVICE* are 0 and 1.

GPU-PACKAGE [CUDA / OPENACC] ~~TINKER8~~

Selects code paths for some GPU algorithms where both CUDA and OpenACC versions have been implemented. The default value is CUDA. Value will be overwritten by environment variable *GPU_PACKAGE*.

6.7 Mathematical Algorithms

RANDOMSEED [integer]

Followed by an integer value, sets the initial seed value for the random number generator used by Tinker. Setting *RANDOMSEED* to the same value as an earlier run will allow exact reproduction of the earlier calculation. (Note that this will not hold across different machine types.) *RANDOMSEED* should be set to a positive integer less than about 2 billion. In the absence of the *RANDOMSEED* keyword the seed is chosen randomly based upon the number of seconds that have elapsed in the current decade.

REFERENCES

[1] Viloya S. Allured, Christine M. Kelly, and Clark R. Landis. SHAPES empirical force field: new treatment of angular potentials and its application to square-planar transition-metal complexes. *Journal of the American Chemical Society*, 113(1):1–12, 1991. doi:10.1021/ja00001a001.

[2] Norman L. Allinger, Young H. Yuh, and Jenn-Huei Lii. Molecular mechanics. The MM3 force field for hydrocarbons. 1. *Journal of the American Chemical Society*, 111(23):8551–8566, 1989. doi:10.1021/ja00205a001.

[3] E. Bright Wilson, Jr., J. C. Decius, and Paul C. Cross. *Molecular Vibrations: The Theory of Infrared and Raman Vibrational Spectra*. Dover Publications, Inc., New York, 1980. ISBN 9780486639413.

[4] Glenn J. Martyna, Mark E. Tuckerman, Douglas J. Tobias, and Michael L. Klein. Explicit reversible integrators for extended systems dynamics. *Molecular Physics*, 87(5):1117–1157, 1996. doi:10.1080/00268979600100761.

[5] Scott E. Feller, Yuhong Zhang, Richard W. Pastor, and Bernard R. Brooks. Constant pressure molecular dynamics simulation: The Langevin piston method. *The Journal of Chemical Physics*, 103(11):4613–4621, 1995. doi:10.1063/1.470648.

[6] Celeste Sagui, Lee G. Pedersen, and Thomas A. Darden. Towards an accurate representation of electrostatics in classical force fields:

Efficient implementation of multipolar interactions in biomolecular simulations. *The Journal of Chemical Physics*, 120(1):73–87, 2004. doi:10.1063/1.1630791.

[7] Wei Wang and Robert D. Skeel. Fast evaluation of polarizable forces. *The Journal of Chemical Physics*, 123(16):164107, 2005. doi:10.1063/1.2056544.

[8] B. T. Thole. Molecular polarizabilities calculated with a modified dipole interaction. *Chemical Physics*, 59(3):341–350, 1981. doi:10.1016/0301-0104(81)85176-2.

INDEX

R

S

T

U