TINKER

Software Tools for Molecular Design
Version 3.7
June 1999

Copyright © 1990-1999 by Jay William Ponder
All Rights Reserved

User's Guide Cover Illustration by Jay Nelson
Courtesy of Prof. R. T. Paine, Univ. of New Mexico

v3.7 6/99

TINKER

Software Tools for Molecular Design
Version 3.7   June 1999

Table of Contents   Page

1.   Introduction to the TINKER Package

Welcome to the TINKER molecular modeling package! TINKER is designed to be an easily used and flexible system of programs and routines for molecular mechanics and dynamics as well as other energy-based and structural manipulation calculations. It is intended to be modular enough to enable development of new computational methods and efficient enough to meet most "production" calculation needs. Rather than incorporating all the functionality in one monolithic program, TINKER provides a set of relatively small programs that interoperate to perform complex computations. New programs can be easily added by modelers with only limited programming experience. The series of major programs included in the distribution system perform the following core tasks:

(1)   energy minimization and structural optimization
(2)   analysis of energy distribution within a structure
(3)   molecular dynamics and stochastic dynamics
(4)   simulated annealing with a choice of cooling schedules
(5)   normal modes and vibrational frequencies
(6)   conformational search and global optimization
(7)   transition state location and conformational pathways
(8)   fitting of energy parameters to crystal data
(9)   distance geometry with pairwise metrization
(10)   molecular volumes and surface areas
(11)   free energy changes for structural mutations
(12)   advanced algorithms based on potential smoothing

Many of the various energy minimization and molecular dynamics computations can be performed on full or partial structures, over Cartesian, internal or rigid body coordinates, and including a variety of boundary conditions and crystal cell types. Other programs are available to generate timing data and allow checking of potential function derivatives for coding errors.

Due to its emphasis on ease of modification, TINKER differs from many other currently available molecular modeling packages in that the user is expected to be willing to write simple "front-end" programs and make some alterations at the source code level. The main programs provided should be considered as templates for the users to change according to their wishes. All subroutines are internally documented and structured programming practices are adhered to throughout. The result, it is hoped, will be a calculational system which can be tailored to local needs and desires.

The core TINKER system consists of about 100,000 lines of source written entirely in a portable Fortran77 dialect. Use is made of only some very common extensions that aid in writing highly structured code. The current version of the package has been ported to a wide range of computers with no or extremely minimal changes. Tested machines include: Compaq Alphas under Tru64 Unix or OpenVMS; Hewlett-Packard, IBM RS/6000, Silicon Graphics and Sun workstations under the vendor's Unix; Apple Macintosh; and Intel PCs under Windows9X, WindowsNT or Linux. At present, our new code is written on Compaq Alpha platforms, and occasionally tested for compatibility on various of the machine and OS combinations listed above. At the present time, we are in the process of converting primary development efforts from FORTRAN77 to both Fortran90 and C/C++. A machine translated C version of TINKER is currently available, and a hand-translated optimized C version in partially complete.

The basic design of the energy function engine used by the TINKER system allows usage of several different parameter sets. At present we are distributing parameters that implement AMBER-95, CHARMM22, MM2, MM3, OPLS-AA, OPLS-UA and our own TINKER parameters. In most cases, the source code separates the geometric manipulations needed for energy derivatives from the actual form of the energy function itself. Several other literature parameter sets are under development (ENCAD, MMFF-94, UFF, *etc.),* and many of the alternative potential function forms reported in the literature can be implemented directly or after minor code changes.

Much of the software in the TINKER package has been heavily used and well tested, but some modules are still in a fairly early stage of development. Further work on the TINKER system is planned in three main areas: (1) extension and improvement of the potential energy parameters including further development of our polarizable multipole TINKER force field, (2) coding of new computational algorithms including additional methods for free energy determination, torsional Monte Carlo and molecular dynamics sampling, advanced methods for long range interactions, better transition state location, and further application of the potential smoothing paradigm, (3) a friendlier user interface for protein/nucleic acid/polysaccharide computations including direct input/output of Protein Data Bank files, and (4) a simple GUI front-end to tie together the TINKER programs using RasMol as a molecule viewer.

Questions and comments regarding the TINKER package, including suggestions for improvements and changes should be made to the author:

Professor Jay William Ponder
Biochemistry & Molecular Biophysics
North Building, Room 2811, Box 8231
Washington University School of Medicine
660 South Euclid Avenue
Saint Louis, MO 63110 U.S.A.

phone: (314) 362-4195
fax: (314) 362-7183
email: ponder@dasher.wustl.edu

In addition, an Internet web site containing an online version of this User's Guide, the most recent distribution version of the full TINKER package and other useful information can be found at http://dasher.wustl.edu/tinker, the Home Page for the TINKER Molecular Modeling Package.

2. Installing TINKER on your Computer

The TINKER package is distributed on the Internet via either the web site or the anonymous ftp account on dasher.wustl.edu with an IP number of 128.252.162.151. This node is an AlphaServer 4100 file and compute server running Compaq Tru64 Unix located in the Ponder lab at Washington University School of Medicine. After pointing to the home page or logging in under ftp (Username: anonymous, Password: "your email address"), the software may be found in the /pub/tinker subdirectory. The package is also available via the Web and standard browsers from the TINKER home page at http://dasher.wustl.edu/tinker/. The complete TINKER distribution as well as individual files can be downloaded from this site.

On dasher.wustl.edu, the TINKER package is present as both a compressed Unix tar archive and as a complete set of uncompressed source and data files. Binaries are provided for Intel PCs running Windows 9X/NT, PCs running Linux, and for Apple Power Macintosh. All of these executables are available in standard compressed formats as individual programs or as complete sets of executables. It is expected that other Unix users and PC users who need specially customized versions, will build binaries for their specific system. Sites with access to the Unix tar and compress/uncompress commands should simply obtain the archive file "tinker.tar.Z". Alternatively, a file "tinker.tar.gz" compressed with GNU gzip is also provided. If you choose to download individual files, you will need at a minimum the contents of the /doc, /source and /params subdirectories. Also required are the compile/build scripts from the subdirectory named for your machine type. Other areas contain test cases and examples, benchmark results, and machine-translated C code. The entire TINKER package, after building the executables, requires about 40 to 60 megabytes of disk space. This value can vary significantly from system to system, and depends in part on the availability and use of shared libraries.

The documentation for the TINKER programs, including the guide you are currently reading, is located in the /pub/tinker/doc subdirectory. The documentation was prepared on Compaq Alpha workstations using the Applixware Words and Graphics programs. Portable versions of the documentation are provided as ascii text in the .txt files and in the .ps Postscript format files. Please read and return the TINKER license. While the intent is to distribute the TINKER code to anyone who wants it, the authors would like to keep track of the sites using the package. The returned license forms also help us justify further development of TINKER. When new modules and capabilities become available, and when the almost inevitable bugs are uncovered, we will attempt to notify all who have returned a license form. Finally, we remind you that this software is copyrighted, and ask that it not be redistributed in any form.

The compilation and building of the TINKER executables should be easy for most of the common workstation and PC class computers. We provide in the /make area a Unix-style Makefile that with some modification can be used to build TINKER on most Unix machines. As a simpler alternative to Makefiles for the Unix versions, we also provide machine-specific directories with three separate shell scripts to compile the source, build an object library, and link binary executables. Three similar command files are provided for Open VMS, and for PC and Macintosh systems. Compilation on Unix workstations should use the vendor supplied Fortran compiler, if available. The public domain GNU g77 Fortran compiler is also capable of building TINKER on Unix machines. TINKER can also be built with the egcs version of the GNU compilers available from http://egcs.cygnus.com/ for Linux and other Unix systems. The Linux executables we provide are built with the Absoft ProFortran for

Linux compiler which generated somewhat faster executables than egcs. For the Macintosh we have favorable experience with the Absoft ProFortran compiler running under the Macintosh Programmers' Workbench (MPW). On IBM PC's running Windows 9X/NT, the distributed TINKER executables are built under the Digital Visual Fortran 6.0 compiler. The Microsoft Fortran Power Station 4.0 and Watcom F77 compilers are also sufficient for building TINKER, and we provide scripts for all three of these PC compilers. Alternative PC compilers such as those from Lahey/Fujitsu and The Portland Group should work as well, but we have not evaluated them yet. Please see the README files in each of the machine-specific areas for further information.

The first step in building TINKER using the script files is to run the appropriate "compile" script. Next you must use the "library" script to create an archive of object code modules. Finally, run the "link" script to produce the complete set of TINKER executables. The executables can be renamed and moved to wherever you like by editing and running the "rename" script.

Regardless of your target machine, only three small pieces of code can possibly require attention prior to building. The first two are the system dependent time and date routines found in "clock.f" and "calendar.f" respectively. Please uncomment the sections of these routines needed for your computer type. The final set of source alterations are the master array dimensions found in the include file "sizes.i". The most basic limit is on the number of atoms allowed, "maxatm". This parameter can be set to 10000 or more on most workstations. Personal computers with minimal memory may need a lower limit, perhaps 1000 atoms, depending on available memory, swap space and other resources. A description of the other parameter values is contained in the header of the file. Note that in order to keep the code completely transparent, TINKER does not implement any sort of virtual memory or heap data structure. This requires that "sizes.i" values be set at least as large as the biggest problem you intend to run. Obviously, you should not set the array sizes to unnecessarily large values, since this can tax your compute resources and may result in performance degradation. The worst case we know of at present is for some of the Alpha machines, where running a "small" problem with TINKER executables dimensioned to "large" sizes can result in a 25-50% CPU time penalty, especially if only the default compiler options are used.

Specific questions about the building or use of the TINKER package should be directed to ponder@dasher.wustl.edu. TINKER related questions or comments of more general interest can be sent to the Computational Chemistry List (http://www.ccl.net/) run by Jan Labanowski of The Ohio Supercomputer Center. The TINKER developers monitor this list and will respond to the list or the individual poster as appropriate.

3.   Types of Input & Output Files

This section describes the basic file types used by the TINKER package. Let's say you wish to perform a calculation on a particular small organic molecule. Assume that the file name chosen for our input and output files is `sample`. Then all of the TINKER files will reside on the computer under the name `sample.xxx` where `.xxx` is any of the several extension types to be described below.

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.INT

The `.int` file contains an internal coordinates representation of the molecular structure. It consists of 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, the force field atom type number of the atom, and internal coordinates in the usual Z-matrix format. For each atom the internal coordinates consist of a distance to some previously defined atom, and either two bond angles or a bond angle and a dihedral angle to previous atoms. The length, angle and dihedral definitions do not have to represent "real" bonded interactions. Following the last atom definition are two optional blank line separated sets of atom number pairs. The first list contains pairs of atoms that are covalently bonded, but whose bond length was not used as part of the atom definitions. These pairs are typically used to close ring structures. The second list contains "bonds" that are to be broken, *i.e.*, pairs of atoms that are not covalently bonded, but which were used to define a distance in the atom definitions.

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 of this User's Guide. 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.DYN

The `.dyn` file contains values needed to restart a molecular or stochastic dynamics computation. It stores the current position, current velocity and current and previous accelerations for each atom, as well as the size and shape of any periodic box or crystal unit cell. This information can be used to start a new dynamics run from the final state of a previous run. Upon startup, the dynamics programs always check for the presence of a `.dyn` file and make use of it whenever possible. The `.dyn` file is updated concurrent with the saving of a new dynamics trajectory snapshot.

SAMPLE.END

The `.end` file type provides a mechanism to gracefully stop a running TINKER calculation. At appropriate checkpoints during a calculation, TINKER will test for the presence of a `sample.end` file, and if found will terminate the calculation after updating the output. The `.end` file can be created at any time during a computation, and will be detected when the next checkpoint is reached. The file may be of zero size, and its contents are unimportant. In the current version of TINKER, the `.end` mechanism is only available within dynamics-based programs.

SAMPLE.001, SAMPLE.002, ...

Several types of computations produce files containing a three or more digit extension (`.001` as shown; or `.002`, `.137`, `.5678`, *etc.*). These are referred to as cycle files, and are used to store various types of output structures. The cycle files from a given computation are identical in internal structure to either the `.xyz` or `.int` files described above. For example, the vibrational analysis program can save the tenth normal mode in `sample.010`. A molecular dynamics-based program might save its tenth 0.1 picosecond frame (or an energy minimizer its tenth partially minimized intermediate) in a file of the same name.

SAMPLE.ARC

A TINKER archive file is simply a series of `.xyz` Cartesian coordinate files appended together one after another. This file can be used to condense the results from intermediate stages of an optimization, frames from a molecular dynamics trajectory, or set of normal mode vibrations into a single file for storage.

SAMPLE.PDB

This file type contains coordinate information in the PDB format developed by the Brookhaven Protein Data Bank for deposition of model structures based on macromolecular X-ray diffraction and NMR data. Although TINKER itself does not use `.pdb` files directly for input/output, auxiliary programs are provided with the system for interconverting `.pdb` files with the `.xyz` format described above.

SAMPLE.SEQ

This file type contains the primary sequence of a biopolymer in a form compatible with the University of Wisconsin GCG package. An `.seq` file is generated automatically when a PDB file is converted to TINKER `.xyz` format, and is required for the reverse conversion.

SAMPLE.FRAC

The fractional coordinates corresponding to the asymmetric unit of a crystal unit cell are stored in the `.frac` file. The internal format of this file is identical to the `.xyz` file; except that the coordinates are fractional instead of in Angstrom units.

SAMPLE.XMOL

The ARCHIVE program has the option of converting a series of `.xyz` cycle files into an XMOL XYZ file. These files can be displayed as a movie using the MSC XMOL display

program. Note that the **.xmol** file format does not contain TINKER atom type information, so it is not possible to convert an **.xmol** file back into a TINKER **.xyz** file.

SAMPLE.BIOS

The ARCHIVE program has the option of converting a series of **.xyz** cycle files into a BIOSYM InsightII coordinate archive file. These files can be displayed as a movie using the InsightII display program. Note that the **.bios** file format does not contain TINKER atom type information, so it is not possible to convert a **.bios** file back into a TINKER **.xyz** file.

PARAMETER FILES

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.

4.   Potential Energy Programs

        This section of the manual contains a brief description of each of the TINKER potential energy programs. A detailed example showing how to run each program is included in a later section.

ALCHEMY

A simple program to perform very basic free energy perturbation calculations. This program is provided mostly for demonstration purposes; a more general and sophisticated version is currently under development in our group. The present version uses the perturbation formula and windowing with an explicit mapping of atoms involved in the mutation ("AMBER"-style), instead of thermodynamic integration and independent freely propagating groups of mutated atoms ("CHARMM"-style). Some of the code specific to this program is limited to OPLS potentials, but could be easily generalized to handle other functional forms. In St. Louis, we use ALCHEMY in molecular modeling course laboratory exercises to perform such classic mutations as chloride to bromide and ethane to methanol in water.

ANALYZE

Provides information about a specific molecular structure. The program will ask for the name of a structure file, which must be in the TINKER `.xyz` file format, and the type of analysis desired. Options allow output of: (1) total potential energy of the system, (2) breakdown of the energy by potential function type or over individual atoms, (3) computation of the total dipole moment and its components, moments of inertia and radius of gyration, (4) listing of the parameters used to compute selected interaction energies, (5) energies associated with specified individual interactions.

ANNEAL

Performs a molecular dynamics simulated annealing computation. The program starts from a specified input molecular structure in TINKER `.xyz` format. The trajectory is updated using either a modified Beeman or a velocity Verlet integration method. The annealing protocol is implemented by allowing smooth changes between starting and final values of the system temperature via the Groningen method of coupling to an external bath. The scaling can be linear or sigmoidal in nature. In addition, parameters such as cutoff distance can be transformed along with the temperature. The user must input the desired number of dynamics steps for both the equilibration and cooling phases, a time interval for the dynamics steps, and an interval between coordinate/trajectory saves. All saved coordinate sets along the trajectory are placed in sequentially numbered cycle files.

DYNAMIC

Performs a molecular dynamics (MD) or stochastic dynamics (SD) computation. Starts either from a specified input molecular structure (an `.xyz` file) or from a structure-velocity-acceleration set saved from a previous dynamics trajectory (a restart from a `.dyn` file). MD trajectories are propagated using either a modified Beeman or a velocity Verlet integration method. SD is implemented via our own derivation of a velocity Verlet-based algorithm. In addition the program can perform full crystal calculations, and can operate in constant energy mode or with maintenance of a desired temperature and/or pressure using the Groningen method of coupling to external baths. The user must input the desired number of

dynamics steps, a time interval for the dynamics steps, and an interval between coordinate/trajectory saves. Coordinate sets along the trajectory can be saved as sequentially numbered cycle files or directly to a TINKER archive **.arc** file. At the same time that a point along the trajectory is saved, the complete information needed to restart the trajectory from that point is updated and stored in the **.dyn** file.

GDA

A program to implement Straub's Gaussian Density Annealing algorithm over an effective series of analytically smoothed potential energy surfaces. This method can be viewed as an extended stochastic version of the diffusion equation method of Scheraga, *et al.*, and also has many similar features to the TINKER Potential Smoothing and Search (PSS) series of programs. The current version of GDA is similar to but does not exactly reproduce Straub's published method and is limited to argon clusters and other simple systems involving only van der Waals interactions; further modification and development of this code is currently underway in the Ponder research group. As with other programs involving potential smoothing, GDA currently requires use of the **smooth.prm** force field parameters.

MINIMIZE

Performs a nonlinear conjugate gradient minimization of an input structure over Cartesian coordinates. The method requires only the potential energy and gradient at each step along the minimization pathway. It requires storage space proportional to the number of atoms in the structure. The MINIMIZE procedure is recommended for preliminary minimization of trial structures to an RMS gradient of 1.0 to 0.1 kcal/mole/Ang. It has a relatively fast cycle time and is tolerant of poor initial structures, but converges very slowly (linearly) once near the minimum. The user supplies the name of the TINKER **.xyz** coordinates file and a target RMS gradient value at which the minimization will terminate. Output consists of minimization statistics written to the screen or redirected to an output file, and the new coordinates written to updated **.xyz** files or to cycle files.

MINIROT

This program uses the same nonlinear conjugate gradient method as MINIMIZE, but performs the computation in terms of dihedral angles instead of Cartesian coordinates. Output is saved in an updated **.int** file or in cycle files.

NEWTON

A truncated Newton minimization method which requires potential energy, gradient and Hessian information. This procedure has significant advantages over standard Newton methods, and is able to minimize very large structures completely. Several options are provided with respect to minimization method and preconditioning of the Newton equations. The default options are recommended unless the user is familiar with the math involved. This program operates in Cartesian coordinate space and is fairly tolerant of poor input structures. Typical algorithm iteration times are longer than with nonlinear conjugate gradient or variable metric methods, but many fewer iterations are required for complete minimization. NEWTON is usually the best choice for minimizations to the 0.01 to 0.000001 kcal/mole/Ang level of RMS gradient convergence. Tests for directions of negative curvature can be removed, allowing NEWTON to be used for optimization to conformational transition state structures (this only works if the starting point is very "close" to the transition state).

Input consists of a TINKER `.xyz` coordinates file; output is an updated set of minimized coordinates and minimization statistics.

NEWTROT

The NEWTROT program is similar to NEWTON except that it requires a `.int` file as input and then operates in terms of dihedral angles as the minimization variables. Since the dihedral space Hessian matrix of an arbitrary structure is often indefinite, this method will often not perform as well as the other, simpler dihedral angle based minimizers.

OPTIMIZE

This is a variable metric energy minimization program which operates on Cartesian coordinates (an `.xyz` file). The method requires computation of energies and gradients as well as storage for an estimate of the Hessian matrix. OPTIMIZE will typically converge somewhat faster and more completely than MINIMIZE. However, the need to store and manipulate a full inverse Hessian estimate limits its use to structures containing less than a few hundred atoms on workstation class machines. As with the other minimizers, OPTIMIZE needs input coordinates and an RMS gradient cutoff criterion. The output coordinates are saved in updated `.xyz` files or as cycle files.

OPTIROT

The OPTIROT program is similar to OPTIMIZE except that it operates on dihedral angles starting from a TINKER `.int` internal coordinate file. This program is usually the preferred method for most dihedral angle optimization problems since Truncated Newton methods appear, in our hands, to lose some of their efficacy in moving from Cartesian to torsional coordinates.

OPTRIGID

The OPTRIGID program is similar to OPTIMIZE except that it operates on rigid bodies starting from a TINKER `.xyz` coordinate file and the rigid body group definitions found in the corresponding `.key` file. Output is saved in an updated `.xyz` file or in cycle files.

PATH

A program that implements a variant of Elber's Lagrangian multiplier-based reaction path following algorithm. The program takes as input a pair of structural minima as TINKER `.xyz` files, and then generates a user specified number of points along a path through conformational space connecting the input structures. The intermediate structures are output as TINKER cycle files, and the higher energy intermediates can be used as input to a Newton-based optimization to locate conformational transition states.

PSS

Implements our version of a potential smoothing and search algorithm for the "global" optimization of molecular conformation. An initial structure in `.xyz` format is first minimized in Cartesian coordinates on a series of increasingly smoothed potential energy surfaces. Then the smoothing procedure is reversed with minimization on each successive surface starting from the coordinates of the minimum on the previous surface. A local search procedure is used during the backtracking to explore for alternative minima better than the

one found during the current minimization. The final result is usually a very low energy conformation or, in favorable cases, the global energy minimum conformation. The minimum energy coordinate sets found on each surface during both the forward smoothing and backtracking procedures are placed in sequentially numbered cycle files.

PSSRIGID

This program implements the potential smoothing and search method as described above for PSS, but performs the computation in terms of keyfile-defined rigid body atom groups instead of Cartesian coordinates. Output is saved in numbered cycle files with the `.xyz` file format.

PSSROT

This program implements the potential smoothing and search method as described above for PSS, but performs the computation in terms of dihedral angles instead of Cartesian coordinates. Output is saved in numbered cycle files with the `.int` file format.

SADDLE

A program for the location of a conformational transition state between two potential energy minima. SADDLE uses a conglomeration of ideas from the Bell-Crighton quadratic path and the Halgren-Lipscomb synchronous transit methods. The basic idea is to perform a nonlinear conjugate gradient optimization in a subspace orthogonal to a suitably defined "reaction coordinate". The program requires as input the coordinates (TINKER `.xyz` files) of the two minima and an RMS gradient convergence criterion for the optimization. The current estimate of the transition state structure is written to the file TSTATE.XYZ. Crude transition state structures generated by SADDLE can sometimes be refined using the NEWTON program. Optionally, a scan of the interconversion pathway can be made at each major iteration.

SCAN

A program for general conformational search of an entire potential energy surface via a basin hopping method. The program takes as input a TINKER `.xyz` coordinates file which is then minimized to find the first local minimum for a search list. A series of activations along various normal modes from this initial minimum are used as seed points for additional minimizations. Whenever a previously unknown local minimum is located it is added to the search list. When all minima on the search list have been subjected to the normal mode activation without locating additional new minima, the program terminates. The individual local minima are written to cycle files as they are discovered. While the SCAN program can be used on standard "undeformed" potential energy surfaces, we have found it to be highly effective at quickly "scanning" a smoothed energy surface to enumerate the major basins of attraction spaning the entire surface.

SNIFFER

A program that implements the Sniffer global optimization algorithm of Butler and Slaminka, a discrete version of Griewank's global search trajectory method. The program takes an input TINKER `.xyz` coordinates file and shakes it vigorously via a modified dynamics trajectory before, hopefully, settling into a low lying minimum. Some trial and error is often required as the current implementation is sensitive to various parameters and

tolerances that govern the computation. At present, these parameters are not user accessible, and must be altered in the source code. However, this method can do a good job of quickly optimizing conformation within a limited range of convergence.

TIMER

A simple program to provide timing statistics for energy function calls within the TINKER package. TIMER requires an input `.xyz` file and outputs the CPU time (wall clock time on some machine types) needed to perform a specified number of energy, gradient and Hessian evaluations.

TIMEROT

This program is similar to TIMER, only it operates over dihedral angles via input of a TINKER `.int` internal coordinate file. In the current version, the torsional Hessian is computed numerically from the analytical torsional gradient.

TESTGRAD

The TESTGRAD program computes and compares the analytical and numerical first derivatives (*i.e.*, the gradient vector) of the potential energy for a Cartesian coordinate input structure. The output can be used to test or debug the current potential or any added user defined energy terms.

TESTHESS

The TESTHESS program computes and compares the analytical and numerical second derivatives (*i.e.*, the Hessian matrix) of the potential energy for a Cartesian coordinate input structure. The output can be used to test or debug the current potential or any added user defined energy terms.

TESTLIGHT

A program to compare the efficiency of different nonbonded neighbor methods for the current molecular system. The program times the computation of energy and gradient for the van der Waals and charge-charge electrostatic potential terms using a simple double loop over all interactions and using the "method of lights" algorithm to select neighbors. The results can be used to decide whether the "method of lights" has any CPU time advantage for the current structure. Both methods should give exactly the same answer in all cases, since the identical individual interactions are computed by both methods. The default double loop method is faster when cutoffs are not used, or when the cutoff sphere contains about half or more of the total system of unit cell. In cases where the cutoff sphere is much smaller than the system size, the "method of lights" can be much faster since it avoids unnecessary calculation of distances beyond the cutoff range.

TESTROT

The TESTROT program computes and compares the analytical and numerical first derivatives (*i.e.*, the gradient vector) of the potential energy with respect to dihedral angles. Input is a TINKER `.int` internal coordinate file. The output can be used to test or debug the current potential or any added user defined energy terms.

VIBRATE

A program to perform vibrational analysis by computing and diagonalizing the full Hessian matrix (*i.e.*, the second partial derivatives) for an input structure (a TINKER `.xyz` file). Eigenvalues and eigenvectors of the mass weighted Hessian (*i.e.*, the vibrational frequencies and normal modes) are also calculated. Structures corresponding to individual normal mode motions can be saved in cycle files.

VIBROT

The program VIBROT forms the torsional Hessian matrix via numerical differentiation of the analytical torsional gradient. The Hessian is then diagonalized and the eigenvalues are output. The present version does not compute the kinetic energy matrix elements needed to convert the Hessian into the torsional normal modes; this will be added in a later version. The required input is a TINKER `.int` internal coordinate file.

XTALFIT

The XTALFIT program is of use in the automated fitting of potential parameters to crystal structure and thermodynamic data. XTALFIT takes as input several crystal structures (TINKER `.xyz` files with unit cell parameters in corresponding keyfiles) as well as information on lattice energies and dipole moments of monomers. The current version uses a nonlinear least squares optimization to fit van der Waals and electrostatic parameters to the input data. Bounds can be placed on the values of the optimization parameters.

XTALMIN

A program to perform full crystal minimizations. The program takes as input the structure coordinates and unit cell lattice parameters. It then alternates cycles of Newton-style optimization of the structure and conjugate gradient optimization of the crystal lattice parameters. This alternating minimization is slower than more direct optimization of all parameters at once, but is somewhat more robust in our hands. The symmetry of the original crystal is not enforced, so interconversion of crystal forms may be observed in some cases.

5.   Structure Manipulation Programs

This section of the manual contains a brief description of each of the TINKER structure manipulation, geometric calculation and auxiliary programs. A detailed example showing how to run each program is included in a later section.

ARCHIVE

A program for concatenating TINKER cycle files into a single archive file; useful for storing the intermediate results of minimizations, dynamics trajectories, and so on. The archive file can be written in TINKER format, or in formats usable with BIOSYM's InsightII (a CAR file) or with MSC's XMOL (their XYZ format). Only active atoms are written the the InsightII and XMOL style files, allowing display of partial structures. The program can also extract individual cycle files from a TINKER archive.

CORRELATE

A program to compute time correlation functions from collections of TINKER cycle files. Its use requires a user supplied function PROPERTY that computes the value of the property for which a time correlation is desired for two input structures. The main body of the program organizes the overall computation in an efficient manner and outputs the final time correlation function.

CRYSTAL

A program for the interconversion of fractional and Cartesian coordinates, generation of the unit cells from an asymmetric unit, and building a "macro" crystal from a single unit cell. The present version can handle about 25 of the most common space groups, others can easily be added as needed by modification of the routine SYMMETRY.

DISTGEOM

A program to perform distance geometry calculations using variations on the classic metric matrix method. A user specified number of structures consistent with keyfile input distance and dihedral restraints is generated. Bond length and angle restraints are derived from the input structure. Trial distances between the triangle smoothed lower and upper bounds can be chosen via any of several metrization schemes. The correct "size" of the structure is automatically maintained by choosing trial distances from Gaussian distributions of appropriate mean and width. The initial embedded structures can be further refined against a geometric restraint-only potential using either a sequential minimization protocol or simulated annealing.

INTEDIT

A program to allow interactive inspection and alteration of the internal coordinate definitions and values of a TINKER structure. If the structure is altered, the user has the option to write out a new internal coordinates file upon exit.

INTXYZ

A program to convert a TINKER `.int` internal coordinates formatted file into a TINKER `.xyz` Cartesian coordinates formatted file.

DOCUMENT

The DOCUMENT program is provided as a minimal listing and documentation tool. It operates on the TINKER source code, either individual files or the complete source listing produced by the command script "listing.make", to generate lists of routines, common blocks or valid keywords. In addition, the program has the ability to output a formatted parameter listing from the standard TINKER parameter files.

PDBXYZ

A program for converting a Brookhaven Protein Data Bank file (a PDB file) into a TINKER `.xyz` Cartesian coordinate file. If the PDB file contains only protein/peptide amino acid residues, then standard protein connectivity is assumed, and transferred to the XYZ file. For non-protein portions of the PDB file, atom connectivity is determined by the program based on interatomic distances. The program also has the ability to add or remove hydrogen atoms from a protein as required by the force field specified during the computation.

PROTEIN

A program for automated building of peptide and protein structures. Upon interactive input of an amino acid sequence with optional phi/psi/omega/chi angles, D/L chirality, etc., the program builds internal and Cartesian coordinates. Standard bond lengths and angles are assumed for the peptide. The program will optionally convert the structure to a cyclic peptide, or add either or both N- and C-terminal capping groups. The final coordinates and a sequence file are produced as the output.

SPACEFILL

A program to compute the volume and surface areas of molecules. Using a modified version of Connolly's original analytical description of the molecular surface, the program determines either the van der Waals, accessible or molecular (contact/reentrant) volume and surface area. Both surface area and volume are broken down into their geometric components, and surface area is decomposed into the convex contribution for each individual atom. The probe radius is input as a user option, and atomic radii can be set via the keyword file.

SUPERPOSE

A program to superimpose two molecular structures in 3-dimensions. A variety of options for input of the atom sets to be used during the superposition are presented interactively to the user. The superposition can be mass-weighted if desired, and the coordinates of the second structure superimposed on the first structure are optionally output.

SYBYLXYZ

A program for converting a TRIPOS Sybyl MOL2 file into a TINKER `.xyz` Cartesian coordinate file. The current version of the program does not attempt to convert the Sybyl atoms types into the active TINKER force field types, *i.e.*, all atoms types are simply set to zero.

XYZINT

A program for converting a TINKER `.xyz` Cartesian coordinate formatted file into a TINKER `.int` internal coordinates formatted file.

XYZEDIT

A program that performs and of a variety of manipulations on an input TINKER .xyz Cartesian coordinates formatted file. The present version of the program has the following interactively selectable options: (1) Renumber All of the Current Atoms, (2) Deletion of Individual Specified Atoms, (3) Deletion of Specified Types of Atoms, (4) Deletion of Atoms outside Cutoff Range, (5) Insertion of Individual Specified Atoms, (6) Replace Old Atom Type with a New Type, (7) Assign Connectivities based on Distance, (8) Convert Units from Bohrs to Angstroms, (9) Invert thru Origin to give Mirror Image, (10) Translate Center of Mass to the Origin, (11) Translate a Specified Atom to the Origin, (12) Create a Periodic Boundary Box, (13) Soak Current Molecule in Box of Solvent, (14) Append another XYZ file to Current One. In most cases, multiply options can be applied sequentially to an input file. At the end of the editing process, a new version of the original `.xyz` file is written as output.

XYZPDB

A program for converting a TINKER `.xyz` Cartesian coordinate file into a Brookhaven Protein Data Bank file (a PDB file).

XYZSYBYL

A program to convert a TINKER `.xyz` Cartesian coordinates file into a TRIPOS Sybyl MOL2 file. The conversion generates only the MOLECULE, ATOM, BOND and SUBSTRUCTURE record type in the MOL2 file. Generic Sybyl atom types are used in most cases; while these atom types may need to be altered in some cases, Sybyl is usually able to correctly display the resulting MOL2 file.

6.   Force Field Parameter Sets

The TINKER package is distributed with several force field parameter sets, implementing a selection of widely-used literature force fields as well as the TINKER force field currently under construction in the Ponder lab. We try to exactly reproduce the intent of the original authors of our distributed, third-party force fields. In all cases the parameter sets have been validated against literature reports, results provided by the original developers, or calculations made with the authentic programs. With the few exceptions noted below, TINKER calculations can be treated as authentic results from the genuine force fields. A brief description of each parameter set, including some still in preparation and not distributed with the current version, is provided below with lead references:

AMBER.PRM

AMBER-95 parameters for proteins. The nucleic acid parameters are not implemented yet. Note that with their "Cornell" force field, the Kollman group has devised separate, fully independent partial charge values for each of the N- and C-terminal residues. At present, the terminal residue charges for TINKER's version maintain the correct formal charge, but redistributed somewhat from the Kollman group values. The file reproduces the authentic parm94 set; torsional parameter changes for parm96 are noted in that section of the file.

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 and 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) [PARM94]

P. Kollman, R. Dixon, W. Cornell, T. Fox, C. Chipot and A. Pohorille, The Development/ Application of a 'Minimalist' Organic/Biochemical Molecular Mechanic Force Field using a Combination of *ab Initio* Calculations and Experimental Data, in **Computer Simulation of Biomolecular Systems**, W. F. van Gunsteren, P. K. Weiner, A. J. Wilkinson, eds., Volume 3, 83-96 (1997) [PARM96]

G. Moyna, H. J. Williams, R. J. Nachman and A. I. Scott, Conformation in Solution and Dynamics of a Structurally Constrained Linear Insect Kinin Pentapeptide Analogue, **Biopolymers**, *49*, 403-413 (1999) [AIB charges]

W. S. Ross and C. C. Hardin, Ion-Induced Stabilization of the G-DNA Quadruplex: Free Energy Perturbation Studies, *J. Am. Chem. Soc.*, *116*, 4363-4366 (1994)   [Alkali Metal Ions]

J. Aqvist, Ion-Water Interaction Potentials Derived from Free Energy Perturbation Simulations, *J. Phys. Chem.*, *94*, 8021-8024, 1990 [Alkaline Earth Ions, radii adapted for AMBER combining rule]

Current parameter values are available from the AMBER site in Peter Kollman's lab at UCSF, http://www.amber.ucsf.edu/amber/amber.html/

CHARMM.PRM

CHARMM22 parameters for proteins. Most of the nucleic acid, lipid and small model compound parameters are not yet implemented.

A. D. MacKerell, Jr., J. Wiorkeiwicz-Kuczera and M. Karplus, An All-Atom Empirical Energy Function for the Simulation of Nucleic Acids, *J. Am. Chem. Soc.*, *117*, 11946-11975 (1995)

A. D. MacKerrell, Jr., *et al.*, All-Atom Empirical Potential for Molecular Modeling and Dynamics Studies of Proteins, *J. Phys. Chem. B*, *102*, 3586-3616 (1998)

S. E. Feller, D. Yin, R. W. Pastor and A. D. MacKerell, Jr., Molecular Dynamics Simulation of Unsaturated Lipids at Low Hydration: Parametrization and Comparison with Diffraction Studies, *Biophysical Journal*, *73*, 2269-2279 (1997) [alkenes]

R. H. Stote and M. Karplus, Zinc Binding in Proteins and Solution - A Simple but Accurate Nonbonded Representation, *Proteins*, *23*, 12-31 (1995) [zinc ion]

Current parameter values are available from the CHARMM parameter site in Alex MacKerell's lab at UMBC, http://www.pharmacy.ab.umd.edu/~alex/

EMR.PRM

Reduced EMR model adapted for "flexible" sidechains. Only a few amino acid residue types have been implemented.

R. V. Pappu, W. J. Schneller and D. L. Weaver, Electrostatic Multipole Representation of a Polypeptide Chain: An Algorithm for Simulation of Polypeptide Properties, *J. Comput. Chem.*, *17*, 1033-1055 (1996)

ENCAD.PRM

ENCAD parameters for proteins and nucleic acids. (*in preparation*)

M. Levitt, M. Hirshberg, R. Sharon and V. Daggett, Potential Energy Function and Parameters for Simulations of the Molecular Dynamics of Protein and Nucleic Acids in Solution, *Comp. Phys. Commun.*, *91*, 215-231 (1995)

HOCH.PRM

Simple NMR-NOE force field of Hoch and Stern.

J. C. Hoch and A. S. Stern, A Method for Determining Overall Protein Fold from NMR Distance Restraints, *J. Biomol. NMR*, *2*, 535-543 (1992)

MERCK.PRM

Preliminary MMFF vdw parameters using buffered 14/7 function.

T. A. Halgren, Representation of van der Waals (vdW) Interactions in Molecular Mechanics Force Fields: Potential Form, Combination Rules, and vdW Parameters, *J. Am. Chem. Soc.*, *114*, 7827-7843 (1992)

MM2.PRM

Full MM2(91) parameters including pi-systems. The anomeric and electronegativity correction terms are not implemented.

N. L. Allinger, "Conformational Analysis. 130. MM2. A Hydrocarbon Force Field Utilizing V1 and V2 Torsional Terms", *J. Am. Chem. Soc.*, *99*, 8127-8134 (1977)

J. T. Sprague, J. C. Tai, Y. Yuh and N. L. Allinger, The MMP2 Calculational Method, *J. Comput. Chem.*, *8*, 581-603 (1987)

N. L. Allinger, R. A. Kok and M. R. Imam, Hydrogen Bonding in MM2, *J. Comput. Chem.*, *9*, 591-595 (1988)

All parameters distributed with TINKER are from the "MM2 (1991) Parameter Set", as provided by N. L. Allinger, University of Georgia

MM3.PRM

Full MM3(99) parameters including pi-systems. The directional hydrogen bonding term is implemented, but the anomeric, electronegativity, Bohlmann correction terms are not implemented.

N. L. Allinger, Y. H. Yuh and J.-H. Lii, Molecular Mechanics. The MM3 Force Field for Hydrocarbons. 1, *J. Am. Chem. Soc.*, *111*, 8551-8566 (1989)

J.-H. Lii and N. L. Allinger, Molecular Mechanics. The MM3 Force Field for Hydrocarbons. 2. Vibrational Frequencies and Thermodynamics, *J. Am. Chem. Soc.*, *111*, 8566-8575 (1989)

J.-H. Lii and N. L. Allinger, Molecular Mechanics. The MM3 Force Field for Hydrocarbons. 3. The van der Waals' Potentials and Crystal Data for Aliphatic and Aromatic Hydrocarbons, *J. Am. Chem. Soc.*, 111, 8576-8582 (1989)

N. L. Allinger, H. J. Geise, W. Pyckhout, L. A. Paquette and J. C. Gallucci, Structures of Norbornane and Dodecahedrane by Molecular Mechanics Calculations (MM3), X-ray Crystallography, and Electron Diffraction, *J. Am. Chem. Soc.*, *111*, 1106-1114 (1989) [torsion-stretch]

N. L. Allinger, F. Li and L. Yan, Molecular Mechanics. The MM3 Force Field for Alkenes, *J. Comput. Chem.*, *11*, 848-867 (1990)

N. L. Allinger, F. Li, L. Yan and J. C. Tai, Molecular Mechanics (MM3) Calculations on Conjugated Hydrocarbons, *J. Comput. Chem.*, *11*, 868-895 (1990)

J.-H. Lii and N. L. Allinger, Directional Hydrogen Bonding in the MM3 Force Field. I, *J. Phys. Org. Chem.*, *7*, 591-609 (1994)

J.-H. Lii and N. L. Allinger, Directional Hydrogen Bonding in the MM3 Force Field. II, *J. Comput. Chem.*, *19*, 1001-1016 (1998)

Current parameter values are available from the MM2/MM3 site in Lou Allinger's lab, http://europa.chem.uga.edu/ccmsd/mm2mm3.html/, and MM3 Parameter Search at http://europa.chem.uga.edu/cgi-bin/mm3para/

All parameters distributed with TINKER were translated from "MM3 PARAMETERS (1999)" as updated on January 17, 1999 and obtained from the above web sites

MM3PRO.PRM

Protein-only version of the MM3 parameters.

J.-H. Lii and N. L. Allinger, The MM3 Force Field for Amides, Polypeptides and Proteins, *J. Comput. Chem.*, *12*, 186-199 (1991)

MMFFPRO.PRM

Protein-only version of the MMFF94 parameters. (*in preparation*)

T. A. Halgren, Merck Molecular Force Field. I. Basis, Form, Scope, Parameterization, and Performance of MMFF94, *J. Comput. Chem.*, *17*, 490-519, 1996

OPLS.PRM

Complete OPLS-UA with united-atom parameters for proteins and many classes of organic molecules. Explicit hydrogens on polar atoms and aromatic carbons.

W. L. Jorgensen and J. Tirado-Rives, The OPLS Potential Functions for Proteins. Energy Minimizations for Crystals of Cyclic Peptides and Crambin, *J. Am. Chem. Soc.*, *110*, 1657-1666 (1988) [Protein & Peptide]

W. L. Jorgensen and D. L. Severance, Aromatic-Aromatic Interactions: Free Energy Profiles for the Benzene Dimer in Water, Chloroform, and Liquid Benzene, *J. Am. Chem. Soc.*, *112*, 4768-4774 (1990) [Aromatic Hydrogens]

S. J. Weiner, P. A. Kollman, D. A. Case, U. C. Singh, C. Ghio, G. Alagona, S. Profeta, Jr. and P. Weiner, A New Force Field for Molecular Mechanical Simulation of Nucleic Acids and Proteins, *J. Am. Chem. Soc.*, *106*, 765-784 (1984) [United-Atom "AMBER/OPLS" Local Geometry]

S. J. Weiner, P. A. Kollman, D. T. Nguyen and D. A. Case, An All Atom Force Field for Simulations of Proteins and Nucleic Acids, *J. Comput. Chem.*, *7*, 230-252 (1986) [All-Atom "AMBER/OPLS" Local Geometry]

L. X. Dang and B. M. Pettitt, Simple Intramolecular Model Potentials for Water, *J. Phys. Chem.*, *91*, 3349-3354 (1987) [Flexible TIP3P and SPC Water]

W. L. Jorgensen, J. D. Madura and C. J. Swenson, Optimized Intermolecular Potential Functions for Liquid Hydrocarbons, *J. Am. Chem. Soc.*, *106*, 6638-6646 (1984) [Hydrocarbons]

W. L. Jorgensen, E. R. Laird, T. B. Nguyen and J. Tirado-Rives, Monte Carlo Simulations of Pure Liquid Substituted Benzenes with OPLS Potential Functions, *J. Comput. Chem.*, *14*, 206-215 (1993) [Substituted Benzenes]

E. M. Duffy, P. J. Kowalczyk and W. L. Jorgensen, Do Denaturants Interact with Aromatic Hydrocarbons in Water?, *J. Am. Chem. Soc.*, *115*, 9271-9275 (1993) [Benzene, Naphthalene, Urea, Guanidinium, TetraMeAmmonium]

W. L. Jorgensen and C. J. Swenson, Optimized Intermolecular Potential Functions for Amides and Peptides. Structure and Properties of Liquid Amides, *J. Am. Chem. Soc.*, *106*, 765-784 (1984) [Amides]

W. L. Jorgensen, J. M. Briggs and M. L. Contreras, Relative Partition Coefficients for Organic Solutes form Fluid Simulations, *J. Phys. Chem.*, *94*, 1683-1686 (1990) [Chloroform, Pyridine, Pyrazine, Pyrimidine]

J. M. Briggs, T. B. Nguyen and W. L. Jorgensen, Monte Carlo Simulations of Liquid Acetic Acid and Methyl Acetate with the OPLS Potential Functions, *J. Phys. Chem.*, *95*, 3315-3322 (1991) [Acetic Acid, Me Acetate]

H. Liu, F. Muller-Plathe and W. F. van Gunsteren, A Force Field for Liquid Dimethyl Sulfoxide and Physical Properties of Liquid Dimethyl Sulfoxide Calculated Using Molecular Dynamics Simulation, *J. Am. Chem. Soc.*, *117*, 4363-4366 (1995) [Dimethyl Sulfoxide]

J. Gao, X. Xia and T. F. George, Importance of Bimolecular Interactions in Developing Empirical Potential Functions for Liquid Ammonia, *J. Phys. Chem.*, *97*, 9241-9246 (1993) [Ammonia]

J. Aqvist, Ion-Water Interaction Potentials Derived from Free Energy Perturbation Simulations, *J. Phys. Chem.*, *94*, 8021-8024 (1990) [Metal Ions]

W. S. Ross and C. C. Hardin, Ion-Induced Stabilization of the G-DNA Quadruplex: Free Energy Perturbation Studies, *J. Am. Chem. Soc.*, *116*, 4363-4366 (1994) [Alkali Metal Ions]

J. Chandrasekhar, D. C. Spellmeyer and 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) [Halide Ions]

Most parameters distributed with TINKER are from "OPLS and OPLS-AA Parameters for Organic Molecules, Ions, and Nucleic Acids" as provided by W. L. Jorgensen, Yale University, October 1997

OPLSAA.PRM

OPLS-AA with all-atom parameters for proteins and many general classes of organic molecules.

W. L. Jorgensen, D. S. Maxwell and 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.*, *117*, 11225-11236 (1996)

W. L. Jorgensen and N. A. McDonald, Development of an All-Atom Force Field for Heterocycles. Properties of Liquid Pyridine and Diazenes, *THEOCHEM-J. Mol. Struct.*, *424*, 145-155 (1998)

N. A. McDonald and 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)

All parameters distributed with TINKER are from "OPLS and OPLS-AA Parameters for Organic Molecules, Ions, and Nucleic Acids" as provided by W. L. Jorgensen, Yale University, October 1997

SMOOTH.PRM

Version of OPLS-UA for use with potential smoothing. Largely adapted largely from standard OPLS-UA parameters with modifications to the vdw and improper torsion terms.

R. V. Pappu, R. K. Hart and J. W. Ponder, Analysis and Application of Potential Energy Smoothing and Search Methods for Global Optimization, *J. Phys, Chem. B*, *102*, 9725-9742 (1998) [Modifications for smoothing]

SMOOTHAA.PRM

Version of OPLS-AA for use with potential smoothing. Largely adapted largely from standard OPLS-AA parameters with modifications to the vdw and improper torsion terms.

R. V. Pappu, R. K. Hart and J. W. Ponder, Analysis and Application of Potential Energy Smoothing and Search Methods for Global Optimization, *J. Phys, Chem. B*, *102*, 9725-9742 (1998) [Modifications for smoothing]

TINKER.PRM

Protein-only parameters for the TINKER force field with multipole values of Dudek and Ponder. The current file contains all the multipole values, but the local geometry and vdw terms are still under development.

WATER.PRM

The current TINKER water parameters for a polarizable multipole electrostatics model. This model is equal or better to the best available water models for many bulk and cluster properties.

Y. Kong and J. W. Ponder, Calculation of the Reaction Field Due to Off-Center Point Multipoles, *J. Chem. Phys.*, *107*, 481-492 (1997)

The parameters distributed with TINKER are from the Ph.D. thesis of Yong Kong, "Multipole Electrostatic Methods for Protein Modeling with Reaction Field Treatment", Biochemistry & Molecular Biophysics, Washington University, St. Louis, August, 1997

7.   Use of the Keyword Control File

This section contains a description of the keyword parameters which may be 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 "ACTIVE 4 -9 17 23" could be used to add atoms 4, 9 through 17, and 23 to the set of active atoms during a TINKER calculation.

Listed below are the valid TINKER keywords sorted into groups by general function. The section ends with an alphabetical listing of the individual keywords along with brief descriptions of their action and possible modifiers, and examples of usage.

Keywords Grouped by Functionality

OUTPUT CONTROL KEYWORDS

ARCHIVE	BINARY	DEBUG
ECHO	EXIT-PAUSE	NOVERSION
OVERWRITE	PRINTOUT	SAVECYCLE
VERBOSE	WRITEOUT	

ENERGY SELECTION KEYWORDS

ANGANGTERM	ANGLETERM	BONDTERM
CHGDPLTERM	CHARGETERM	DIPOLETERM
EWALDTERM	EXTRATERM	IMPROPERTERM

IMPTORSTERM	MPOLETERM	OPBENDTERM
POLARIZETERM	RESTRAINTERM	RXNFIELDTERM
SOLVATETERM	STRBNDTERM	STRTORTERM
TORSIONTERM	TORTORTERM	UREYTERM
VDWTERM		

ENERGY PARAMETER KEYWORDS

ANGANG	ANGLE	ANGLE3
ANGLE4	ANGLE5	ANGLEF
ATOM	BIOTYPE	BOND
BOND3	BOND4	BOND5
CHARGE	DIPOLE	DIPOLE3
DIPOLE4	DIPOLE5	HBOND
IMPROPER	IMPTORS	MULTIPOLE
OPBEND	PIATOM	PIBOND
POLARIZE	SOLVATE	STRBND
STRTORS	TORSION	TORSION4
TORSION5	UREYBRAD	VDW
VDW14	VDWPR	

ENERGY FUNCTIONAL FORM KEYWORDS

A-EXPTERM	B-EXPTERM	C-EXPTERM
ANGLE-CUBIC	ANGLE-QUARTIC	ANGLE-PENTIC
ANGLE-SEXTIC	ANGLEUNIT	ANGANGUNIT
BOND-CUBIC	BOND-QUARTIC	BONDTYPE
BONDUNIT	CHG-12-USE	CHG-13-USE
CHG-14-USE	CHG-SCALE	DIELECTRIC
EPSILONRULE	FORCEFIELD	GAUSSTYPE
OPBENDUNIT	PARAMETERS	PISYSTEM
POLAR-DAMP	POLAR-EPS	POLARIZATION
RADIUSRULE	RADIUSTYPE	RADIUSSIZE
REACTIONFIELD	STRBNDUNIT	STRTORUNIT
TORSIONUNIT	UREYUNIT	VDW-12-USE
VDW-13-USE	VDW-14-USE	VDW-SCALE
VDWTYPE		

POTENTIAL FUNCTION CUTOFF KEYWORDS

CHG-CUTOFF	CHG-TAPER	CUTOFF
DPL-CUTOFF	DPL-TAPER	GROUP-NEIGHBORS
HESS-CUTOFF	LIGHTS	NEUTRAL-GROUPS
TAPER	TRUNCATE	VDW-CUTOFF
VDW-TAPER		

EWALD SUMMATION KEYWORDS

| EWALD-ALPHA | EWALD-CUTOFF | PME-GRID |
| PME-ORDER | | |

CRYSTAL LATTICE & PERIODIC BOUNDARY KEYWORDS

A-AXIS	B-AXIS	C-AXIS
ALPHA	BETA	GAMMA
OCTAHEDRON	SPACEGROUP	

OPTIMIZATION KEYWORDS

FCTMIN	NEXTITER	ANGMAX
CAPPA	HGUESS	EPSLN
FAST	SLOW	PERIOD
NEWHESS	MAXITER	INTMAX
STEPMAX	STEPMIN	STEEPEST-DESCENT
FLETCHER-REEVES	POLAK-RIBIERE	HESTENES-STIEFEL
POWELL-BEALE		

DYNAMICS KEYWORDS

| COMPRESS | FRICTION | INTEGRATE |
| TAU-PRESSURE | TAU-TEMPERATURE | |

TRANSITION STATE KEYWORDS

| DIVERGE | GAMMAMIN | REDUCE |
| SADDLEPOINT | | |

DISTANCE GEOMETRY KEYWORDS

| TRIAL-DISTANCE | TRIAL-DISTRIBUTION |

RANDOM NUMBER KEYWORDS

RANDOMSEED

FREE ENERGY PERTURBATION KEYWORDS

| LAMBDA | MUTATE |

PARTIAL STRUCTURE KEYWORDS

| ACTIVE | GROUP | INACTIVE |
| SELECT-GROUP | | |

CONSTRAINT & RESTRAINT KEYWORDS

BASIN	RATTLE	RATTLE-BOND
RESTRAIN-DIHEDRAL	RESTRAIN-DISTANCE	RESTRAIN-POSITION
SPHERE	WALL	

POTENTIAL SMOOTHING KEYWORDS

| DEFORM | DIFFUSE-ANGLE | DIFFUSE-BOND |
| DIFFUSE-CHARGE | DIFFUSE-IMPROPER | DIFFUSE-TORSION |

DIFFUSE-VDW

Description of Individual Keywords

The following is an alphabetical list of the TINKER keywords along with a brief description of the action of each keyword and required or optional parameters that can be used to extend or modify each keyword. The form of possible modifiers, if any, is shown in brackets following each keyword.

A-AXIS [real]   Sets the value of the a-axis length for a crystal unit cell, or, equivalently, the X-axis length for a periodic box. The length value in Angstroms is listed after the keyword.

A-EXPTERM [real]   Sets the value of the "A" premultiplier term in the Buckingham van der Waals function, *i.e.*, the value of A in the formula $E_{vdw} = \varepsilon \{ A \exp[-B(R_0/R)] - C (R_0/R)^6 \}$.

ACTIVE [integer list]   Sets the list of active atoms during a TINKER computation. Individual potential energy terms are computed when at least one atom involved in the term is active. For Cartesian space calculations, active atoms are those allowed to move. For torsional space calculations, rotations are allowed when all atoms on one side of the rotated bond are active. Multiple ACTIVE lines can be present in the keyfile and are treated cumulatively. On each line the keyword can be followed by one or more atom numbers or atom ranges. The presence of any ACTIVE keyword overrides any INACTIVE keywords in the keyfile.

ALPHA [real]   Sets the value of the α angle of a crystal unit cell, *i.e.*, the angle between the b-axis and c-axis of a unit cell, or, equivalently, the angle between the Y-axis and Z-axis of a periodic box. The default value in the absence of the ALPHA keyword is 90 degrees.

ANGANG [1 integer & 3 reals]   This keyword provides the values for a single angle-angle cross term potential parameter.

ANGANGTERM [NONE/ONLY]   This keyword controls use of the angle-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.

ANGANGUNIT [real]   Sets the scale factor needed to convert the energy value computed by the angle-angle cross term potential into units of kcal/mole. The correct value is force field dependent and typically provided in the header of the master force field parameter file. The default of $(\pi/180)^2 = 0.0003046$ is used, if the ANGANGUNIT keyword is not given in the force field parameter file or the keyfile.

ANGLE [3 integers & 4 reals]   This keyword 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/mole/radian^2, but this can be controlled via the ANGLEUNIT keyword.

ANGLE-CUBIC [real]   Sets the value 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-PENTIC [real]   Sets the value 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-QUARTIC [real]   Sets the value 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-SEXTIC [real]   Sets the value 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.

ANGLE3 [3 integers & 4 reals]   This keyword provides the values for a single bond angle bending parameter specific to atoms in 3-membered rings. 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. 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. The default units for the force constant are kcal/mole/radian^2, but this can be controlled via the ANGLEUNIT keyword. If any ANGLE3 keywords are present, either in the master force field parameter file or the keyfile, then TINKER requires that special ANGLE3 parameters be given for all angles in 3-membered rings. In the absence of any ANGLE3 keywords, standard ANGLE parameters will be used for bonds in 3-membered rings.

ANGLE4 [3 integers & 4 reals]   This keyword provides the values for a single bond angle bending parameter specific to atoms in 4-membered rings. 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. 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. The default units for the force constant are kcal/mole/radian^2, but this can be controlled via the ANGLEUNIT keyword. If any ANGLE4 keywords are present, either in the master force field parameter file or the keyfile, then TINKER requires that special ANGLE4 parameters be given for all angles in 4-membered rings. In the absence of any ANGLE4 keywords, standard ANGLE parameters will be used for bonds in 4-membered rings.

ANGLE5 [3 integers & 4 reals]   This keyword provides the values for a single bond angle bending parameter specific to atoms in 5-membered rings. 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. 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. The default units for the force constant are kcal/mole/radian^2, but this can be controlled via the ANGLEUNIT keyword. If any ANGLE5 keywords are present, either in the master force field parameter file or the keyfile, then TINKER requires that special ANGLE5 parameters be given for all angles in 5-membered rings. In the absence of any ANGLE5 keywords, standard ANGLE parameters will be used for bonds in 5-membered rings.

ANGLEF [3 integers & 3 reals]   This keyword 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/mole/radian^2, but this can be controlled via the ANGLEUNIT keyword.

ANGLETERM [NONE/ONLY]   This keyword 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/mole. The correct value is force field dependent and typically provided in the header of the master force field parameter file. The default value of $(\pi/180)^2 = 0.0003046$ is used, if the ANGLEUNIT keyword is not given in the force field parameter file or the keyfile.

ANGMAX [real]   Set the maximum permissible angle between the current optimization search direction and the negative of the gradient direction. If this maximum angle value is exceeded, the optimization routine will note an error condition and may restart from the steepest descent direction. The default value in the absence of the ANGMAX keyword is usually 88 degrees for conjugate gradient methods and 180 degrees (*i.e.*, disabled) for variable metric optimizations.

ARCHIVE   Causes TINKER molecular dynamics-based programs to write trajectories directly to a single archive file with the `.arc` format. If an archive file already exists at the start of the calculation, then the newly generated trajectory is appended to the end of the existing file. The default in the absence of this keyword is to write the trajectory snapshots to consecutively numbered cycle files.

ATOM [2 integers, name, quoted string, integer, real & integer]   This keyword provides the values needed to define a single force field atom type.

B-AXIS [real]   Sets the value of the b-axis length for a crystal unit cell, or, equivalently, the Y-axis length for a periodic box. The length value in Angstroms is listed after the keyword. If the keyword is absent, the b-axis length is set equal to the a-axis length.

B-EXPTERM [real]   Sets the value of the "B" exponential factor in the Buckingham van der Waals function, *i.e.*, the value of B in the formula $E_{vdw} = \varepsilon \{ A \exp[-B(R_0/R)] - C (R_0/R)^6 \}$.

BASIN [2 reals]   Presence of this keyword turns on a "basin" restraint potential function that serves to drive the system toward a compact structure. The actual function is an exponential of the form $E_{basin} = \Sigma A \exp[-B R]$, summed over all pairs of atoms where R is the distance between atoms. The A and B values are the depth and width parameters given as modifiers to the BASIN keyword. This potential is currently used to control the degree of expansion during potential energy smooth procedures through the use of shallow, broad basins.

BETA [real]   Sets the value of the β angle of a crystal unit cell, *i.e.*, the angle between the a-axis and c-axis of a unit cell, or, equivalently, the angle between the X-axis and Z-axis of a periodic box. The default value in the absence of the BETA keyword is to set the β angle equal to the α angle as given by the keyword ALPHA.

BINARY   Causes TINKER molecular dynamics-based programs to write trajectories directly to a single archive file with a binary format. The default in the absence of this keyword is to write the trajectory to consecutively numbered cycle files or archive files in plain text ASCII format. ***Note this keyword is present in the code, but not active in TINKER version 3.7.***

BIOTYPE [integer, name, quoted string & integer]   This keyword provides the values to define the correspondence between a single biopolymer atom type and its force field atom type.

BOND [2 integers & 2 reals]   This keyword 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 for the bond and the ideal bond length in Å. The default units for the force constant are kcal/mole/Å^2, but this can be controlled via the BONDUNIT keyword.

BOND-CUBIC [real]   Sets the value 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 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.

BOND3 [2 integers & 2 reals]   This keyword provides the values for a single bond stretching parameter specific to atoms in 3-membered rings. 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 for the bond and the ideal bond length in Å. The default units for the force constant are kcal/mole/Å^2, but this can be controlled via the BONDUNIT keyword. If any BOND3 keywords are present, either in the master force field parameter file or the keyfile, then TINKER requires that special BOND3 parameters be given for all bonds in 3-membered rings. In the absence of any BOND3 keywords, standard BOND parameters will be used for bonds in 3-membered rings.

BOND4 [2 integers & 2 reals]   This keyword provides the values for a single bond stretching parameter specific to atoms in 4-membered rings. 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 for the bond and the ideal bond length in Å. The default units for the force constant are kcal/mole/Å^2, but this can be controlled via the BONDUNIT keyword. If any BOND4 keywords are present, either in the master force field parameter file or the keyfile, then TINKER requires that special BOND4 parameters be given for all bonds in 4-membered rings. In the absence of any BOND4 keywords, standard BOND parameters will be used for bonds in 4-membered rings

BOND5 [2 integers & 2 reals]   This keyword provides the values for a single bond stretching parameter specific to atoms in 5-membered rings. 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 for the bond and the ideal bond length in Å. The default units for the force constant are kcal/mole/Å^2, but this can be controlled via the BONDUNIT keyword. If any BOND5 keywords are present, either in the master force field parameter file or the keyfile, then TINKER requires that special BOND5 parameters be given for all bonds in 5-membered rings. In the absence of any BOND5 keywords, standard BOND parameters will be used for bonds in 5-membered rings

BONDTERM [NONE/ONLY]   This keyword 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.

BONDTYPE [TAYLOR/MORSE/GAUSSIAN]   Chooses the functional form of the bond stretching potential. The TAYLOR 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 GAUSSIAN uses an inverted Gaussian with amplitude equal to the Morse "bond dissociation energy" and width set to reproduce the vibrational frequency of a harmonic potential. The default is to use the TAYLOR potential.

BONDUNIT [real]   Sets the scale factor needed to convert the energy value computed by the bond stretching potential into units of kcal/mole. 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.

C-AXIS [real]   Sets the value of the C-axis length for a crystal unit cell, or, equivalently, the Z-axis length for a periodic box. The length value in Angstroms is listed after the keyword. If the keyword is absent, the C-axis length is set equal to the A-axis length.

C-EXPTERM [real]   Sets the value of the "C" dispersion multipler in the Buckingham van der Waals function, *i.e.*, the value of C in the formula $E_{vdw} = \varepsilon \{ A \exp[-B(R_0/R)] - C (R_0/R)^6 \}$.

CAPPA [real]   This keyword is used to set the normal termination criterion for the line search phase of TINKER optimization routines. The line search exits successfully if the ratio of the current gradient projection on the line to the projection at the start of the line search falls below the value of CAPPA. A default value of 0.1 is used in the absence of the CAPPA keyword.

CHARGE [1 integer & 1 real]   This keyword provides a value for a single atomic partial charge electrostatic parameter. The integer modifier, if positive, gives the atom type number for which the charge parameter is to be defined. Note that vdw parameters are given for atom types, not atom classes. If the integer modifier is negative, then the parameter value to follow applies only to the individual atom whose atom number is the negative of the modifier. The real number modifier gives the values of the atomic partial charge in electrons.

CHARGETERM [NONE/ONLY]   This keyword controls use of the charge-charge potential energy term between pairs of atomic partial charges. 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.

CHG-12-USE [NONE]   This keyword determines whether the electrostatic potential energy terms will include interactions between 1-2 connected atoms, *i.e.*, directly bonded atoms. In the absence of a modifying option, this keyword turns on 1-2 interactions reduced by a multiplicative factor supplied via the CHG-SCALE keyword. The NONE option turns off use of 1-2 interactions. The default in the absence of the CHG-12-USE keyword is to omit these interactions from the energy evaluation.

CHG-13-USE [NONE]   This keyword determines whether the electrostatic potential energy terms will include interactions between 1-3 connected atoms, *i.e.*, atoms separated by 2 covalent bonds. In the absence of a modifying option, this keyword turns on 1-3 interactions reduced by a multiplicative factor supplied via the CHG-SCALE keyword. The NONE option turns off use of 1-3 interactions. The default in the absence of the CHG-13-USE keyword is to omit these interactions from the energy evaluation.

CHG-14-USE [NONE]   This keyword determines whether the electrostatic potential energy terms will include interactions between 1-4 connected atoms, *i.e.*, atoms separated by 3 covalent bonds. In the absence of a modifying option, this keyword turns on 1-4 interactions reduced by a multiplicative factor supplied via the CHG-SCALE keyword. The NONE option turns off use of 1-4 interactions. The default in the absence of the CHG-14-USE keyword is to include these interactions in the energy evaluation.

CHG-CUTOFF [real]   Sets the cutoff distance value in Angstroms for simple charge-charge electrostatics and/or for atomic multipole potential energy interactions. The energy for any pair of sites beyond the cutoff distance will be set to zero. Other keywords can be used to select a smoothing scheme near the cutoff distance. The default cutoff distance in the absence of the CHG-CUTOFF keyword is essentially infinite for nonperiodic systems and 10.0 for periodic systems.

CHG-SCALE [real]   This keyword provides a scale factor by which charge-charge electrostatic interactions are reduced between neighboring atoms. The value associated with the CHG-SCALE keyword may be applied to 1-2, 1-3 and/or 1-4 connected atom pairs as specified via the CHG-12-USE, CHG-13-USE and CHG-14-USE keywords. The default value of 1.0 is used, if the CHG-SCALE keyword is not given in either the parameter file or the keyfile.

CHG-TAPER [real]   This keyword allows modification of the cutoff windows for simple charge-charge electrostatics and/or for atomic multipole potential energy interactions. It is similar in form and action to the TAPER keyword, except that its value applies only to the simple charge and multipole potentials. The default value in the absence of the CHG-TAPER keyword is to begin the cutoff window at 0.65 of the corresponding cutoff distance.

CHGDPLTERM [NONE/ONLY]   This keyword controls use of the charge-dipole potential energy term between atomic partial charges and bond dipoles. 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.

COMPRESS [real]   Sets the value of the bulk solvent isothermal compressibility in Atm^{-1} for use during pressure computation and scaling in molecular dynamics computations. The default value used in the absence of the COMPRESS keyword is 0.000046, appropriate for water. This parameter serves as a scale factor for the Groningen-style pressure bath coupling time, and its exact value should not be of critical importance.

CUTOFF [real]   Sets the cutoff distance value for all nonbonded potential energy interactions. The energy for any of the nonbonded potentials of a pair of sites beyond the cutoff distance will be set to zero. Other keywords can be used to select a smoothing scheme near the cutoff distance, or to apply different cutoff distances to various nonbonded energy terms.

DEBUG   Turns on printing of detailed information and intermediate values throughout the progress of a TINKER computation; not recommended for use with large structures or full potential energy functions since a summary of every individual interaction will usually be output.

DEFORM [real]   Sets the amount of diffusion equation-style smoothing that will be applied to the potential energy surface when using the SMOOTH force field. The real number option is equivalent to the "time" value in the original Piela, *et al.* formalism; the larger the value, the greater the smoothing. The default value is zero, meaning that no smoothing will be applied.

DIELECTRIC [real]   Sets the value of the bulk dielectric constant used to damp all electrostatic interaction energies for any of the TINKER electrostatic potential functions.

The default value is force field dependent, but is usually equal to 1.0 (for Allinger's MM force fields the default is 1.5).

DIFFUSE-ANGLE [real]   This keyword is used during potential function smoothing procedures to specify the effective diffusion coefficient to be applied to the smoothed form of the bond angle bending potential function. In the absence of the DIFFUSE-ANGLE keyword, a default value of 0.0014 is used.

DIFFUSE-BOND [real]   This keyword is used during potential function smoothing procedures to specify the effective diffusion coefficient to be applied to the smoothed form of the bond stretching potential function. In the absence of the DIFFUSE-BOND keyword, a default value of 0.000156 is used.

DIFFUSE-CHARGE [real]   This keyword is used during potential function smoothing procedures to specify the effective diffusion coefficient to be applied to the smoothed form of the Coulomb's Law charge-charge potential function. In the absence of the DIFFUSE-CHARGE keyword, a default value of 3.5 is used.

DIFFUSE-IMPROPER [real]   This keyword is used during potential function smoothing procedures to specify the effective diffusion coefficient to be applied to the Gaussian form of the improper dihedral potential function. In the absence of the DIFFUSE-IMPROPER keyword, a default value of 0.0225 is used.

DIFFUSE-TORSION [real]   This keyword is used during potential function smoothing procedures to specify the effective diffusion coefficient to be applied to the smoothed form of the torsion angle potential function. In the absence of the DIFFUSE-TORSION keyword, a default value of 0.0225 is used.

DIFFUSE-VDW [real]   This keyword is used during potential function smoothing procedures to specify the effective diffusion coefficient to be applied to the smoothed Gaussian approximation to the Lennard-Jones van der Waals potential function. In the absence of the DIFFUSE-VDW keyword, a default value of 1.0 is used.

DIPOLE [2 integers & 2 reals]   This keyword provides the values for a single bond dipole electrostatic parameter. The integer modifiers give the atom type numbers for the two kinds of atoms involved in the bond dipole which is to be defined. The real number modifiers give the value of the bond dipole in Debyes and the position of the dipole site along the bond. If the bond dipole value is positive, then the first of the two atom types is the positive end of the dipole. For a negative bond dipole value, the first atom type listed is negative. The position along the bond is an optional modifier that gives the postion of the dipole site as a fraction between the first atom type (position=0) and the second atom type (position=1). The default for the dipole position in the absence of a specified value is 0.5, placing the dipole at the midpoint of the bond.

DIPOLE3 [2 integers & 2 reals]   This keyword provides the values for a single bond dipole electrostatic parameter specific to atoms in 3-membered rings. The integer modifiers give the atom type numbers for the two kinds of atoms involved in the bond dipole which is to be defined. The real number modifiers give the value of the bond dipole in Debyes and the position of the dipole site along the bond. The default for the dipole position in the absence of a specified value is 0.5, placing the dipole at the midpoint of the bond. If any DIPOLE3 keywords are present, either in the master force field parameter file or the keyfile, then TINKER requires that special DIPOLE3 parameters be given for all bond dipoles in 3-

membered rings. In the absence of any DIPOLE3 keywords, standard DIPOLE parameters will be used for bonds in 3-membered rings.

DIPOLE4 [2 integers & 2 reals]   This keyword provides the values for a single bond dipole electrostatic parameter specific to atoms in 4-membered rings. The integer modifiers give the atom type numbers for the two kinds of atoms involved in the bond dipole which is to be defined. The real number modifiers give the value of the bond dipole in Debyes and the position of the dipole site along the bond. The default for the dipole position in the absence of a specified value is 0.5, placing the dipole at the midpoint of the bond. If any DIPOLE4 keywords are present, either in the master force field parameter file or the keyfile, then TINKER requires that special DIPOLE4 parameters be given for all bond dipoles in 4-membered rings. In the absence of any DIPOLE4 keywords, standard DIPOLE parameters will be used for bonds in 4-membered rings.

DIPOLE5 [2 integers & 2 reals]   This keyword provides the values for a single bond dipole electrostatic parameter specific to atoms in 5-membered rings. The integer modifiers give the atom type numbers for the two kinds of atoms involved in the bond dipole which is to be defined. The real number modifiers give the value of the bond dipole in Debyes and the position of the dipole site along the bond. The default for the dipole position in the absence of a specified value is 0.5, placing the dipole at the midpoint of the bond. If any DIPOLE5 keywords are present, either in the master force field parameter file or the keyfile, then TINKER requires that special DIPOLE5 parameters be given for all bond dipoles in 5-membered rings. In the absence of any DIPOLE5 keywords, standard DIPOLE parameters will be used for bonds in 5-membered rings.

DIPOLETERM [NONE/ONLY]   This keyword controls use of the dipole-dipole potential energy term between pairs of bond dipoles. 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.

DIVERGE [real]   This keyword is used by the SADDLE program to set the maximum allowed value of the ratio of the gradient length along the path to the total gradient norm at the end of a cycle of minimization perpendicular to the path. If the value provided by the DIVERGE keyword is exceeded, then another cycle of maximization along the path is required. A default value of 0.005 is used in the absence of the DIVERGE keyword.

DPL-CUTOFF [real]   Sets the cutoff distance value in Angstroms for bond dipole-bond dipole electrostatic potential energy interactions. The energy for any pair of bond dipole sites beyond the cutoff distance will be set to zero. Other keywords can be used to select a smoothing scheme near the cutoff distance. The default cutoff distance in the absence of the DPL-CUTOFF keyword is essentially infinite for nonperiodic systems and 10.0 for periodic systems.

DPL-TAPER [real]   This keyword allows modification of the cutoff windows for bond dipole-bond dipole electrostatic potential energy interactions. It is similar in form and action to the TAPER keyword, except that its value applies only to the vdw potential. The default value in the absence of the DPL-TAPER keyword is to begin the cutoff window at 0.75 of the dipole cutoff distance.

ECHO [text string]   The presence of this keyword causes whatever text follows it on the line to be copied directly to the output file. This keyword is also active in parameter files. It

has no default value; if no text follows the ECHO keyword, a blank line is placed in the output file.

EPSILONRULE [GEOMETRIC/ARITHMETIC/HARMONIC/HHG]   This keyword selects the combining rule used to derive the ε value for van der Waals interactions. The default in the absence of the EPSILONRULE keyword is to use the GEOMETRIC mean of the individual ε values of the two atoms involved in the van der Waals interaction.

EPSLN [real]   This keyword specifies the minimum size of a move in the optimization parameters that will be accepted during conjugate gradient optimization. If the size of the move for a single iteration is less than this value, an error condition is noted that may lead to termination of the optimization. The default value in the absence of the EPSLN keyword should be the machine precision and is set to 10^{-16} in the double precision version of TINKER.

EWALD-ALPHA [real]   Sets the value of the Ewald coefficient which controls the width of the Gaussian screening charges during particle mesh Ewald summation. In the absence of the EWALD-ALPHA keyword, a value is chosen which causes interactions outside the real-space cutoff to be below a fixed tolerance. For most standard applications of PME, the program default should be used.

EWALD-CUTOFF [real]   Sets the value in Angstroms of the real-space distance cutoff for use during particle mesh Ewald summation. By default, in the absence of the EWALD-CUTOFF keyword, a value of 9.0 is used.

EWALDTERM [NONE/ONLY]   This keyword controls use of particle mesh Ewald summation to compute the potential energy term between atomic partial charges under periodic boundary conditions. In the absence of a modifying option, this keyword turns on use PME when applicable. The NONE option turns off use of PME, and thus the charge-charge energy term. The ONLY option turns off all potential energy terms except for the PME-based charge-charge energy. Note that PME is used by default for partial charge electrostatics under periodic boundary conditions. To use spherical cutoff methods with periodic boundaries, it is necessary to turn off use of PME and enable the charge-charge term with the CHARGETERM keyword.

EXIT-PAUSE   This keyword causes TINKER programs to pause and wait for a carriage return at the end of execution prior to returning control to the operating system. This is useful to keep the execution window open following termination on machines running Microsoft Windows or Apple MacOS. The default in the absence of the EXIT-PAUSE keyword, is to return control to the operating system immediately at program termination.

EXTRATERM [NONE/ONLY]   This keyword controls use of the user defined extra 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.

FAST [real]   This keyword specifies a rapid rate of function decrease during conjugate gradient optimization. If the ratio of the objective function decrease on the current iteration to the amount remaining above the desired function value as specified by the FCTMIN keyword is greater than the value specified by FAST, then the next step will be taken in the steepest descent direction. The default value in the absence of the FAST keyword is 0.5.

FCTMIN [real]   This keyword sets a convergence criterion for successful completion of a TINKER optimization. If the value of the optimization objective function, typically the potential energy, falls below the value set by FCTMIN, then the optimization is deemed to have converged. The default value in the absence of the FCTMIN keyword is -1000000, effectively removing this criterion as a possible agent for termination.

FLETCHER-REEVES   The presence of this keyword causes the conjugate gradient nonlinear optimization routine employed by MINIMIZE, MINIROT and other programs to use the Fletcher-Reeves update formula instead of the default Memoryless Quasi-Newton update formula.

FORCEFIELD [name]   This keyword provides a name for the force field to be used in the current calculation. Its value is usually set in the master force field parameter file for the calculation (see the PARAMETERS keyword) instead of in the keyfile.

FRICTION [real]   Sets the value of the frictional coefficient in ps^{-1} for use with stochastic dynamics. The default value used in the absence of the FRICTION keyword is 91.0, which is generally appropriate for water.

GAMMA [real]   Sets the value of the γ angle of a crystal unit cell, *i.e.*, the angle between the a-axis and b-axis of a unit cell, or, equivalently, the angle between the X-axis and Z-axis of a periodic box. The default value in the absence of the GAMMA keyword is to set the γ angle equal to the α angle as given by the keyword ALPHA.

GAMMAMIN [real]   Sets the convergence target value for γ during searches for maxima along the quadratic synchronous transit used by the SADDLE program. The value of γ is the square of the ratio of the gradient projection along the path to the total gradient. A default value of 0.00001 is used in the absence of the GAMMAMIN keyword.

GAUSSTYPE [LJ-2/LJ-4/MM2-2/MM3-2/IN-PLACE]   This keyword specifies the underlying vdw form that a Gaussian vdw approximation will attempt to fit.number of terms to be used in a Gaussian approximation of the Lennard-Jones van der Waals potential. The text modifier gives the name of the functional form to be used. Thus LJ-2 as a modifier will result in a 2-Gaussian fit to a Lennard-Jones vdw potential. The GAUSSTYPE keyword only takes effect when VDWTYPE is set to GAUSSIAN. This keyword has no default value.

GROUP [integer, integer list]   This keyword defines an atom group as a substructure within the full input molecular structure. The value of the first integer is the group number which must be in the range from 1 to the maximum number of allowed groups. The remaining intergers give the atom or atoms contained in this group as one or more atom numbers or ranges. Multiple keyword lines can be used to specify additional atoms in the same group. Note that an atom can only be in one group, the last group to which it is assigned is the one used.

GROUP-NEIGHBORS   This keyword causes the attached atom to be used in determining the charge-charge neighbor distance for all monovalent atoms in the molecular system. Its use causes all monovalent atoms to be treated the same as their attached atoms for purposes of including or scaling 1-2, 1-3 and 1-4 interactions. This option works only for the simple "charge-charge" electrostatic potential; it does not affect bond dipole or atomic multipole potentials. The GROUP-NEIGHBORS scheme is similar to that used by some common force fields such as ENCAD.

HBOND [2 integers & 2 reals]   This keyword provides the values for the MM3-style directional hydrogen bonding parameters for a single pair of atoms. The integer modifiers give the pair of atom class numbers for which hydrogen bonding parameters are to be defined. The two real number modifiers give the values of the minimum energy contact distance in Å and the well depth at the minimum distance in kcal/mole.

HESS-CUTOFF [real]   This keyword defines a lower limit for significant Hessian matrix elements. During computation of the Hessian matrix of partial second derivatives, any matrix elements with absolute value below HESS-CUTOFF will be set to zero and omitted from the sparse matrix Hessian storage scheme used by TINKER. For most calculations, the default in the absence of this keyword is zero, i.e., all elements will be stored. For most Truncated Newton optimizations the Hessian cutoff will be chosen dynamically by the optimizer.

HESTENES-STIEFEL   The presence of this keyword causes the conjugate gradient nonlinear optimization routine employed by MINIMIZE, MINIROT and other programs to use the Hestenes-Stiefel update formula instead of the default Memoryless Quasi-Newton update formula.

HGUESS [real]   Sets an initial guess for the average value of the diagonal elements of the scaled inverse Hessian matrix. This value is used only by the optimally conditioned variable metric optimization routine in TINKER. A default value of 0.4 is used in the absence of the HGUESS keyword.

IMPROPER [4 integers & 2 reals]   This keyword provides the values for a single CHARMM-style improper torsional angle parameter.

IMPROPERTERM [NONE/ONLY]   This keyword controls use of the CHARMM-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.

IMPTORS [4 integers & up to 3 real/real/integer triples]   This keyword 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, phase offset in degrees and periodicity of a particular improper torsional term, respectively. Periodicities through 3-fold are allowed for improper torsional parameters.

IMPTORSTERM [NONE/ONLY]   This keyword 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.

INACTIVE [integer list]   Sets the list of inactive atoms during a TINKER computation. Individual potential energy terms are not computed when all atoms involved in the term are inactive. For Cartesian space calculations, inactive atoms are not allowed to move. For torsional space calculations, rotations are not allowed when there are inactive atoms on both sides of the rotated bond. Multiple INACTIVE lines can be present in the keyfile, and on

each line the keyword can be followed by one or more atom numbers or ranges. If any INACTIVE keys are found, all atoms are set to active except those listed on the INACTIVE lines. The ACTIVE keyword overrides all INACTIVE keywords found in the keyfile.

INTEGRATE [VERLET/BEEMAN/STOCHASTIC]   Chooses the integration method for propagation of dynamics trajectories. The keyword is followed on the same line by the name of the option. Standard Newtonian MD can be run using either VERLET for the Velocity Verlet method, or BEEMAN for the velocity form of Bernie Brook's "Better Beeman" method. A Velocity Verlet-based stochastic dynamics trajectory is selected by the STOCHASTIC modifier. The default integration scheme is MD using the BEEMAN method.

INTMAX [integer]   Sets the maximum number of interpolation cycles that will be allowed during the line search phase of an optimization. All gradient-based TINKER optimization routines use a common line search routine involving quadratic extrapolation and cubic interpolation. If the value of INTMAX is reached, an error status is set for the line search and the search is repeated with a much smaller initial step size. The default value in the absence of this keyword is optimization routine dependent, but is usually in the range 5 to 10.

LAMBDA [real]   This keyword sets the value of the λ path parameter for free energy perturbation calculations. The real number modifier specifies the position along the mutation path and must be a number in the range from 0 (initial state) to 1 (final state). The actual atoms involved in the mutation are given separately in individual MUTATE keyword lines.

LIGHTS   This keyword turns on Method of Lights neighbor generation for the charge-charge potential and any of the van der Waals potentials. This method will yield identical energetic results to the standard double loop method. "Lights" will be faster when the volume of a sphere with radius equal to the nonbond cutoff distance is significantly less than half the volume of the total system (*i.e.*, the full molecular system, the crystal unit cell or the periodic box).

MAXITER [integer]   Sets the maximum number of minimization iterations that will be allowed for any TINKER program that uses any of the nonlinear optimization routines. The default value in the absence of this keyword is program dependent, but is always set to a very large number.

MPOLETERM [NONE/ONLY]   This keyword controls use of the atomic multipole electrostatics 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.

MULTIPOLE [5 lines with: 3 integers & 1 real, 3 reals, 1 real, 2 reals, 3 reals]   This keyword provides the values for a single set of atomic multipole parameters.

MUTATE [3 integers]   This keyword is used to specify atoms to be mutated during free energy perturbation calculations. The first integer modifier gives the atom number of an atom in the current system. The final two modifier values give the atom types corresponding the the λ=0 and λ=1 states of the specified atom.

NEUTRAL-GROUPS   The keyword causes the attached atom to be used in determining the charge-charge interaction cutoff distance for all monovalent atoms in the molecular system. Its use reduces cutoff discontinuities by avoiding splitting many of the largest

"charge separations" found in typical molecules. Note that this keyword does not rigorously implement the usual concept of a "neutral group" as used in the literature with AMBER/OPLS and other force fields. This option works only for the simple "charge-charge" electrostatic potential; it does not affect bond dipole or atomic multipole potentials.

NEWHESS [integer]   Sets the number of algorithmic iterations between recomputation of the Hessian matrix. At present this keyword applies exclusively to optimizations using the Truncated Newton method. The default value in the absence of this keyword is 1, *i.e.,* the Hessian is computed on every iteration.

NEXTITER [integer]   Sets the iteration number to be used for the first iteration of the current computation. At present this keyword applies to optimization procedures where its use can effect convergence criteria, timing of restarts, and so forth. The default in the absence of this keyword is to take the initial iteration as iteration 1.

NOVERSION   Turns off the use of "version numbers" appended to the end of filenames as the method for generating filenames for updated copies of an existing file. The presence of this keyword results in direct use of input file names without a search for the highest available version, and requires the entry of specific output file names in many additional cases. By default, in the absence of this keyword, TINKER generates and attaches version numbers in a manner similar to the Digital OpenVMS operating system. For example, subsequent new versions of the file `molecule.xyz` would be written first to the file `molecule.xyz_2`, then to `molecule.xyz_3`, *etc.*

OCTAHEDRON   Specifies that the periodic "box" is a truncated octahedron with maximal distance across the truncated octahedron as given by the A-AXIS keyword. All other unit cell and periodic box size-defining keywords are ignored if the OCTAHEDRON keyword is present.

OPBEND [2 integers & 1 real]   This keyword provides the values for a single Allinger MM-style out-of-plane angle bending potential parameter.

OPBENDTERM [NONE/ONLY]   This keyword controls use of the Allinger MM-style 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 Allinger MM-style out-of-plane bending potential into units of kcal/mole. The correct value is force field dependent and typically provided in the header of the master force field parameter file. The default of $(\pi/180)^2 = 0.0003046$ is used, if the OPBENDUNIT keyword is not given in the force field parameter file or the keyfile.

OVERWRITE   Causes TINKER programs, such as minimizations, that output intermediate coordinate sets to create a single disk file for the intermediate results which is successively overwritten with the new intermediate coordinates as they become available. This keyword is essentially the opposite of the SAVECYCLE keyword.

PARAMETERS [file name]   Provides the name of the force field parameter file to be used for the current TINKER calculation. The standard file name extension for parameter files, `.prm`, is an optional part of the file name modifier. The default in the absence of the PARAMETERS keyword is to look for a parameter file with the same base name as the

molecular system and ending in the `.prm` extension. If a valid parameter file is not found, the user will asked to provide a file name interactively.

PERIOD [integer]   This keyword specifies the maximum number of conjugate gradient minimization iterations between restarts, *i.e.*, zeroing the accumulated conjugate directions and taking a step in the steepest descent direction. The default is to perform a periodic restart every 200 iterations or after a number of iterations equal to the number of variables being optimized, whichever is greater.

PIATOM [1 integer & 3 reals]   This keyword provides the values for the pisystem MO potential parameters for a single atom class belonging to a pisystem.

PIBOND [2 integers & 2 reals]   This keyword provides the values for the pisystem MO potential parameters for a single type of pisystem bond.

PISYSTEM [integer list]   This keyword sets the atoms within a molecule that are part of a conjugated π-system. The keyword is followed on the same line by a list of atom numbers and/or atom ranges that constitute the π-system. The Allinger MM force fields use this information to set up an MO calculation used to scale bond and torsion parameters involving π-system atoms.

PME-GRID [3 integers]   This keyword sets the dimensions of the charge grid used during particle mesh Ewald summation. The three modifiers give the size along the X-, Y- and Z-axes, respectively. If either the Y- or Z-axis dimensions are omitted, then they are set equal to the X-axis dimension. The default in the absence of the PME-GRID keyword is to set the grid size along each axis to the smallest power of 2, 3 and/or 5 which is at least as large as 1.5 times the axis length in Angstoms. Note that the FFT used by PME is not restricted to, but is most efficient for, grid sizes which are powers of 2, 3 and/or 5.

PME-ORDER [integer]   This keyword sets the order of the B-spline interpolation used during particle mesh Ewald summation. A default value of 8 is used in the absence of the PME-ORDER keyword.

POLAK-RIBIERE   The presence of this keyword causes the conjugate gradient nonlinear optimization routine employed by MINIMIZE, MINIROT and other programs to use the Polak-Ribiere update formula instead of the default Memoryless Quasi-Newton update formula.

POLAR-DAMP [2 reals]   Controls the strength of the damping function applied to induced dipoles and dipole polarization interaction energies. The first modifier sets the radius in Angstoms of a hypothetical atom with unit polarizability, while the second modifier sets the scale factor for the exponent of the damping function. The default values for the radius and the scale factor are 1.662 and 1.0, respectively. Damping is eliminated entirely by using this keyword to set the radius value to zero.

POLAR-EPS [real]   This keyword sets the convergence criterion applied during computation of self-consistent induced dipoles. The calculation is deemed to have converged when the RMS change (in Debyes) of the induced dipoles at all polarizable sites is less than the value specified with this keyword. The default value in the absence of the keyword is 10^{-6} Debyes.

POLARIZATION [DIRECT/MUTUAL]   Selects between the use of direct and mutual dipole polarization for force fields that incorporate the polarization term. The DIRECT modifier avoids an iterative calculation by using only the permanent electric field in computation of induced dipoles. The MUTUAL option, which is the default in the absence of the POLARIZATION keyword, iterates the induced dipoles to self-consistency.

POLARIZE [1 integer & 1 real]   This keyword provides the values for a single atomic dipole polarizability parameter. The integer modifier, if positive, gives the atom type number for which a polarizability parameter is to be defined. If the integer modifier is negative, then the parameter value to follow applies only to the individual atom whose atom number is the negative of the modifier. The real number modifier gives the value of the dipole polarizability in Å^3.

POLARIZETERM [NONE/ONLY]   This keyword controls use of the atomic dipole polarization 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.

POWELL-BEALE   The presence of this keyword causes the conjugate gradient nonlinear optimization routine employed by MINIMIZE, MINIROT and other programs to use the Powell-Beale update formula and restart method instead of the default Memoryless Quasi-Newton update formula.

PRINTOUT [integer]   A general parameter for iterative procedures such as minimizations that sets the number of iterations between writes of status information to the standard output. The default value in the absence of the keyword is 1, *i.e.*, the calculation status is given every iteration.

RADIUSRULE [ARITHMETIC/GEOMETRIC/CUBIC-MEAN]   Sets the functional form of the radius combining rule for heteroatomic van der Waals potential energy interactions. The default in the absence of the RADIUSRULE keyword is to use the arithmetic mean combining rule to get radii for heteroatomic interactions.

RADIUSSIZE [RADIUS/DIAMETER]   Determines whether the atom size values given in van der Waals parameters read from VDW keyword statements are interpreted as atomic radius or diameter values. The default in the absence of the RADIUSSIZE keyword is to assume that vdw size parameters are given as radius values.

RADIUSTYPE [R-MIN/SIGMA]   Determines whether atom size values given in van der Waals parameters read from VDW keyword statements are interpreted as potential minimum (R_{min}) or LJ-style sigma (σ) values. The default in the absence of the RADIUSTYPE keyword is to assume that vdw size parameters are given as R_{min} values.

RANDOMSEED [integer]   Followed by an integer value, this keyword 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.

RATTLE [BONDS/ANGLES/DIATOMIC/TRIATOMIC/WATER]   Invokes the rattle algorithm, a velocity version of shake, on portions of a molecular system during a molecular dynamic calculation. The RATTLE keyword can be followed by any of the modifiers shown, in which case all occurrences of the modifier species are constrained at "ideal" values taken from the bond and angle parameters of the force field in use. In the absence of any modifier, RATTLE constrains all bonds to hydrogen atoms at ideal bond lengths.

RATTLE-BOND [2 integers]   This keyword allows the use of rattle (see above) on a bond between the two atoms whose numbers are specified on the keyword line. If the two atoms are involved in a covalent bond, then their distance is constrained to the "ideal" bond length from the force field. For nonbonded atoms, the rattle constraint fixes their distance at the distance in the input coordinate file.

REACTIONFIELD [2 reals & 1 integer]   This keyword provides parameters needed for the reaction field potential energy calculation. The two real modifiers give the radius of the dielectric cavity and the ratio of the bulk dielectric outside the cavity to that inside the cavity. The integer modifier gives the number of terms in the reaction field summation to be used. In the absence of the REACTIONFIELD keyword, the default values are a cavity of radius 1000000 Å, a dielectric ratio of 80 and use of only the first term of the reaction field summation.

REDUCE [real]   Specifies the fraction between zero and one by which the path between starting and final conformational state will be shortened at each major cycle of the transition state location algorithm implemented by the SADDLE program. This causes the path endpoints to move up and out of the terminal structures toward the transition state region. In favorable cases, a nonzero value of the REDUCE modifier can speed convergence to the transition state. The default value in the absence of the REDUCE keyword is zero.

RESTRAIN-DIHEDRAL [4 integers & 3 reals]   This keyword implements a flat-welled harmonic potential that can be used to restrain the dihedral angle between four atoms to lie within a specified angle range. The initial integer modifiers contains the atom numbers of the four atoms whose dihedral angle, computed in the atom order listed, is to be restrained. The first two real number modifiers give the lower and upper bounds in degrees on the allowed dihedral angle values. The angle values given can wrap around across -180 and +180 degrees. Outside the allowed angle range, a harmonic potential with force constant in kcal/degree^2 given by the final real modifier is applied. If the force constant is omitted, a default value of 1.0 is used. If all the real modifiers are omitted, then the atoms are restrained to a dihedral angle of zero with the default force constant.

RESTRAIN-DISTANCE [2 integers & 3 reals]   This keyword implements a flat-welled harmonic potential that can be used to restrain two atoms to lie within a specified distance range. The initial integer modifiers contains the atom numbers of the two atoms to be restrained. If the interatomic distance lies between the lower and upper bounds, the restraint potential is zero. Outside the bounds, a harmonic potential with force constant in kcal/Å^2 given by the final real modifier is applied. If the force constant is omitted, a default value of 100.0 is used. If all the real modifiers are omitted, then the atoms are restrained to an interatomic distance of zero with the default force constant.

RESTRAIN-POSITION [1 integer & 4 reals]   This keyword provides the ability to restrain an individual atom to a specified coordinate position. The initial 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 final real modifier sets

the force constant in kcal/Å^2 for the harmonic restraint potential. If the force constant is omitted, a default value of 100.0 is used. If all the real modifiers are omitted, then the atom is restrained to the origin with the default force constant.

RESTRAINTERM [NONE/ONLY]   This keyword 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.

RXNFIELDTERM [NONE/ONLY]   This keyword controls use of the reaction field continuum solvation 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.

SADDLEPOINT   The presence of this keyword allows Newton-style second derivative-based optimization routine used by NEWTON, NEWTROT and other programs to converge to saddlepoints as well as minima on the potential surface. By default, in the absence of the SADDLEPOINT keyword, checks are applied that prevent convergence to stationary points having directions of negative curvature.

SAVECYCLE   Causes TINKER programs, such as minimizations, that output intermediate coordinate sets to save each successive set to the next consecutively numbered cycle file. This keyword is essentially the opposite of the OVERWRITE keyword.

SELECT-GROUP [2 integers, real]   This keyword gives the weight in the final potential energy of a specified set of intra- or intergroup interactions. The integer modifiers give the group numbers of the groups involved. If the two numbers are the same, then an intragroup set of interactions is specified. The real modifier gives the weight by which all energetic interactions in this set will be multiplied before incorporation into the final potential energy. If omitted as a keyword modifier, the weight will be set to 1.0 by default. If any SELECT-GROUP keywords are present, then any set of interactions not specified in a SELECT-GROUP keyword is given a zero weight. The default when no SELECT-GROUP keywords are specified is to use all intergroup interactions with a weight of 1.0 and to set all intragroup interactions to zero.

SLOW [real]   This keyword specifies the limit of an unacceptably slow rate of function decrease during conjugate gradient optimization. If the ratio of the objective function decrease on the current iteration to the amount remaining above the desired function value as specified by the FCTMIN keyword is less than the value specified by SLOW, then an error condition is noted that may lead to termination. The default value in the absence of the SLOW keyword is zero, effectively removing this progress check.

SOLVATE [ASP/SASA/GBSA integer]   Use of this keyword during energy calculations with any of the standard force fields turns on a macroscopic hydration free energy term. The modifier selects between the Eisenberg-McLachlan ASP method (using the Wesson-Eisenberg vacuum-to-water parameters), the Ooi-Scheraga SASA method, or the Macromodel GB/SA method. In the absence of a modifier, the ASP method is used. At present, GB/SA is only valid for force fields that use simple partial charge electrostatics. The GBSA modifier takes an additional integer value that is the maximum number of atoms for which the original Macromodel algorithm is used to compute Born radii. If a system has more than this number of atoms, the approximate GB/SA method of Hawkins, Cramer and Truhlar is used.

The default value for the maximum number of atoms for the Macromodel GB/SA Born radii is 300.

SOLVATETERM [NONE/ONLY]   This keyword controls use of the macroscopic solvation 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.

SPACEGROUP [name]   This keyword selects the space group to be used in manipulation of crystal unit cells and asymmetric units. The name option must be chosen from one of the following currently implemented space groups: P1, P1(-), P21, Cc, P21/a, P21/n, P21/c, C2/c, P212121, Pna21, Pn21a, Cmc21, Pccn, Pbcn, Pbca, P41, I41/a, P4(-)21c, P4(-)m2, R3c, P6(3)/mcm, Fm3(-)m, Im3(-)m.

SPHERE [4 reals, or 1 integer & 1 real]   This keyword provides an alternative to the ACTIVE and INACTIVE keywords for specification of subsets of active atoms. If four real number modifiers are provided, the first three are taken as X-, Y- and Z-coordinates and the fourth is the radius of a sphere centered at these coordinates. In this case, all atoms within the sphere at the *start* of the calculation are active *throughout* the calculation, while all other atoms are inactive. Similarly if one integer and real number are given, an "active" sphere with radius set by the real is centered on the system atom with atom number given by the integer modifier. Multiple SPHERE keyword lines can be present in a single keyfile, and the list of active atoms specified by the spheres is cumulative.

STEEPEST-DESCENT   The presence of this keyword causes the conjugate gradient nonlinear optimization routine employed by MINIMIZE, MINIROT and other programs to use the simple steepest descent update formula instead of the default Memoryless Quasi-Newton update formula.

STEPMAX [real]   This keyword and its modifying value set the maximum size of an individual step during the line search phase of conjugate gradient or truncated Newon optimizations. The step size is computed as the norm of the vector of changes in parameters being optimized. The default value depends on the particular TINKER program, but is usually in the range from 1.0 to 5.0 when not specified via the STEPMAX keyword.

STEPMIN [real]   This keyword and its modifying value set the minimum size of an individual step during the line search phase of conjugate gradient or truncated Newon optimizations. The step size is computed as the norm of the vector of changes in parameters being optimized. The default value is usually set to about 10^{-16} when not specified via the STEPMIN keyword.

STRBND [1 integer & 3 reals]   This keyword provides the values for a single stretch-bend cross term potential parameter. The integer modifier gives the atom class number for the central atom of the bond angle involved in stretch-bend interactions. The real number modifiers give the force constant values to be used when the central atom of the angle is attached to 0, 1 or 2 additional hydrogen atoms, respectively. The default units for the stretch-bend force constant are kcal/mole/Å-degree, but this can be controlled via the STRBNDUNIT keyword.

STRBNDTERM [NONE/ONLY]   This keyword 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/mole. 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 STRBNDUNIT keyword is not given in the force field parameter file or the keyfile.

STRTORS [2 integers & 1 real]   This keyword 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 central bond atom classes. The default units for the stretch-torsion force constant can be controlled via the STRTORUNIT keyword.

STRTORTERM [NONE/ONLY]   This keyword 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/mole. 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.

TAPER [real]   This keyword allows modification of the cutoff windows for nonbonded potential energy interactions. The nonbonded terms are smoothly reduced from their standard value at the beginning of the cutoff window to zero at the far end of the window. The far end of the window is specified via the CUTOFF keyword or its potential function specific variants. The modifier value supplied with the TAPER keyword sets the beginning of the cutoff window. The modifier can be given either as an absolute distance value in Angstroms, or as a fraction between zero and one of the CUTOFF distance. The default value in the absence of the TAPER keyword ranges from 0.65 to 0.9 of the CUTOFF distance depending on the type of potential function. The windows are implemented via polynomial-based switching functions, in some cases combined with energy shifting.

TAU-PRESSURE [real]   Sets the coupling time in picoseconds for the Groningen-style pressure bath coupling used to control the system pressure during molecular dynamics calculations. A default value of 2.0 is used for TAU-PRESSURE in the absence of the keyword.

TAU-TEMPERATURE [real]   Sets the coupling time in picoseconds for the Groningen-style temperature bath coupling used to control the system temperature during molecular dynamics calculations. A default value of 0.1 is used for TAU-TEMPERATURE in the absence of the keyword.

TORSION [4 integers & up to 6 real/real/integer triples]   This keyword 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 half-amplitude, phase offset in

degrees and periodicity of a particular torsional function term, respectively. Periodicities through 6-fold are allowed for torsional parameters.

TORSION4 [4 integers & up to 6 real/real/integer triples]   This keyword provides the values for a single torsional angle parameter specific to atoms in 4-membered rings. The first four integer modifiers give the atom class numbers for the atoms involved in the torsional angle to be defined. The remaining triples of real number and integer modifiers operate as described above for the TORSION keyword.

TORSION5 [4 integers & up to 6 real/real/integer triples]   This keyword provides the values for a single torsional angle parameter specific to atoms in 5-membered rings. The first four integer modifiers give the atom class numbers for the atoms involved in the torsional angle to be defined. The remaining triples of real number and integer modifiers operate as described above for the TORSION keyword.

TORSIONTERM [NONE/ONLY]   This keyword 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/mole. 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.

TORTORTERM [NONE/ONLY]   This keyword 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. This energy term is not implemented in the current version of TINKER.

TRIAL-DISTANCE [CLASSIC/RANDOM/TRICOR/HAVEL integer/PAIRWISE integer]   Sets the method for selection of a trial distance matrix during distance geometry computations. The keyword takes a modifier that selects the method to be used. The HAVEL and PAIRWISE modifiers also require an additional integer value that specifies the number of atoms used in metrization and the percentage of metrization, respectively. The default in the absence of this keyword is to use the PAIRWISE method with 100 percent metrization. Further information on the various methods is given with the description of the TINKER distance geometry program.

TRIAL-DISTRIBUTION [real]   Sets the initial value for the mean of the Gaussian distribution used to select trial distances between the lower and upper bounds during distance geometry computations. The value given must be between 0 and 1 which represent the lower and upper bounds respectively. This keyword is rarely needed since TINKER will usually be able to choose a reasonable value by default.

TRUNCATE   Causes all distance-based nonbond energy cutoffs to be sharply truncated to an energy of zero at distances greater than the value set by the cutoff keyword(s) without use of any shifting, switching or smoothing schemes. At all distances within the cutoff sphere, the full interaction energy is computed.

UREYBRAD [3 integers & 2 reals]   This keyword 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 Å. The default units for the force constant are kcal/mole/Å^2, but this can be controlled via the UREYUNIT keyword.

UREYTERM [NONE/ONLY]   This keyword 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/mole. 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.

VDW [1 integer & 3 reals]   This keyword provides values for a single van der Waals parameter. The integer modifier, if positive, gives the atom class number for which vdw parameters are to be defined. Note that vdw parameters are given for atom classes, not atom types. The three real number modifiers give the values of the atom size in Å, homoatomic well depth in kcal/mole, and an optional reduction factor for univalent atoms.

VDW-12-USE [NONE]   This keyword determines whether the van der Waals potential energy will include interactions between 1-2 connected atoms, *i.e.*, directly bonded atoms. In the absence of a modifying option, this keyword turns on 1-2 interactions reduced by a multiplicative factor supplied via the VDW-SCALE keyword. The NONE option turns off use of 1-2 interactions. The default in the absence of the VDW-12-USE keyword is to omit these interactions from the energy evaluation.

VDW-13-USE [NONE]   This keyword determines whether the van der Waals potential energy will include interactions between 1-3 connected atoms, *i.e.*, atoms separated by 2 covalent bonds. In the absence of a modifying option, this keyword turns on 1-3 interactions reduced by a multiplicative factor supplied via the VDW-SCALE keyword. The NONE option turns off use of 1-3 interactions. The default in the absence of the VDW-13-USE keyword is to omit these interactions from the energy evaluation.

VDW-14-USE [NONE]   This keyword determines whether the van der Waals potential energy will include interactions between 1-4 connected atoms, *i.e.*, atoms separated by 3 covalent bonds. In the absence of a modifying option, this keyword turns on 1-4 interactions reduced by a multiplicative factor supplied via the VDW-SCALE keyword. The NONE option turns off use of 1-4 interactions. The default in the absence of the VDW-14-USE keyword is to include these interactions in the energy evaluation.

VDW-CUTOFF [real]   Sets the cutoff distance value in Angstroms for van der Waals potential energy interactions. The energy for any pair of van der Waals sites beyond the cutoff distance will be set to zero. Other keywords can be used to select a smoothing scheme near the cutoff distance. The default cutoff distance in the absence of the VDW-CUTOFF keyword is essentially infinite for nonperiodic systems and 10.0 for periodic systems.

VDW-SCALE [real]   This keyword provides a scale factor by which van der Waals potential energy interactions are reduced between neighboring atoms. The value associated with the VDW-SCALE keyword may be applied to 1-2, 1-3 and/or 1-4 connected atom pairs as specified via the VDW-12-USE, VDW-13-USE and VDW-14-USE keywords. The default value of 1.0 is used, if the VDW-SCALE keyword is not given in either the parameter file or the keyfile.

VDW-TAPER [real]   This keyword allows modification of the cutoff windows for van der Waals potential energy interactions. It is similar in form and action to the TAPER keyword, except that its value applies only to the vdw potential. The default value in the absence of the VDW-TAPER keyword is to begin the cutoff window at 0.9 of the vdw cutoff distance.

VDW14 [1 integer & 2 reals]   This keyword provides values for a single van der Waals parameter for use in 1-4 nonbonded interactions. The integer modifier, if positive, gives the atom class number for which vdw parameters are to be defined. Note that vdw parameters are given for atom classes, not atom types. The two real number modifiers give the values of the atom size in Å and the homoatomic well depth in kcal/mole. Reduction factors, if used, are carried over from the VDW keyword for the same atom class.

VDWPR [2 integers & 2 reals]   This keyword provides the values for the vdw parameters for a single special heteroatomic pair of atoms. The integer modifiers give the pair of atom class numbers for which special vdw parameters are to be defined. The two real number modifiers give the values of the minimum energy contact distance in Å and the well depth at the minimum distance in kcal/mole.

VDWTERM [NONE/ONLY]   This keyword controls use of the van der Waals repulsion-dispersion 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.

VDWTYPE [LENNARD-JONES/BUCKINGHAM/BUFFERED-14-7/MM3-HBOND /GAUSSIAN]   Sets the functional form for the van der Waals potential energy term. The text modifier gives the name of the functional form to be used. The GAUSSIAN modifier value implements a two or four Gaussian fit to the corresponding Lennard-Jones function for use with potential energy smoothing schemes. The default in the absence of the VDWTYPE keyword is to use the standard two parameter Lennard-Jones function.

VERBOSE   Turns on printing of secondary and informational output during a variety of TINKER computations; a subset of the more extensive output provided by the DEBUG keyword.

WALL [real]   Sets the radius of a spherical boundary used to maintain "droplet" boundary conditions. The real modifier specifies the desired approximate radius of the droplet. In practice, an artificial van der Waals wall is constructed at a fixed buffer distance of 2.5 Å outside the specified radius. The effect is that atoms which attempt to move outside the region defined by the droplet radius will be forced toward the center.

WRITEOUT [integer]   A general parameter for iterative procedures such as minimizations that sets the number of iterations between writes of intermediate results (such as the current coordinates) to disk file(s). The default value in the absence of the keyword is 1, *i.e.*, the intermediate results are written to file on every iteration. Whether

successive results are saved to new files or replace previously written intermediate results is controlled by the OVERWRITE and SAVECYCLE keywords.

8. <u>Notes on Special Features & Methods</u>

This section contains several short notes with further information about TINKER methodology, algorithms and special features. The discussion is not intended to be exhaustive, but rather to explain features and capabilities so that users can make more complete use of the package.

FILE VERSION NUMBERS

All of the input and output file types routinely used by the TINKER package are capable of existing as multiple versions of a base file name. For example, if the program XYZINT is run on the input file `molecule.xyz`, the output internal coordinates file will be written to `molecule.int`. If a file named `molecule.int` is already present prior to running XYZINT, then the output will be written instead to the next available version, in this case to `molecule.int_2`. In fact the output is generally written to the lowest available, previously unused version number (`molecule.int_3`, `molecule.int_4`, *etc.*, as high as needed). Input file names are handled similarly. If simply `molecule` or `molecule.xyz` is entered as the input file name upon running XYZINT, then the highest version of `molecule.xyz` will be used as the actual input file. If an explicit version number is entered as part of the input file name, then the specified version will be used as the input file.

The version number scheme will be recognized by many older users as a holdover from the VMS origins of the first version of the TINKER software. It has been maintained to make it easier to chain together multiple calculations that may create several new versions of a given file, and to make it more difficult to accidently overwrite a needed result. The version scheme applies to most uses of many common TINKER file types such as `.xyz`, `.int`, `.key`, `.arc`. It is not used when an overwritten file "update" is obviously the correct action, for example, the `.dyn` molecular dynamics restart files. For those users who prefer a more "Unix-like" operation, and do not desire use of file versions, this feature can be turned off by adding the NOVERSION keyword to the applicable TINKER keyfile.

The version scheme as implemented in TINKER does have two known quirks. First, it becomes impossible to directly use the original "unversioned" copy of a file if higher version numbers are present. For example, if the files `molecule.xyz` and `molecule.xyz_2` both exist, then `molecule.xyz` cannot be accessed as input by XYZINT. If `molecule.xyz` is entered in response to the input file name question, `molecule.xyz_2` (or the highest present version number) will be used as input. The only workaround is to copy or rename `molecule.xyz` to something else, say `molecule.new`, and use that name for the input file. Secondly, missing version numbers always end the search for the highest available version number; *i.e.*, version numbers are assumed to be consecutive and without gaps. For example, if `molecule.xyz`, `molecule.xyz_2` and `molecule.xyz_4` are present, but not `molecule.xyz_3`, then `molecule.xyz_2` will be used as input to XYZINT if `molecule` is given as the input file name. Similarly, output files will fill in gaps in an already existing set of file versions.

COMMAND LINE OPTIONS

Many operating systems or compiler supplied-libraries make available something like the standard Unix iargc and getarg routines for capturing command line arguments. On these machines most of the TINKER programs support a selection of command line arguments and options. The name of the keyfile to be used for a calculation is read from the

argument following a **-k** (equivalent to either **-key** or **-keyfile**, case insensitive) command line argument. Note that the **-k** options can appear anywhere on the command line following the executable name. All other command line arguments, excepting the name of the executable program itself, are treated as input arguments. These input arguments are read from left to right and interpreted in order as the answers to questions that would be asked by an interactive invocation of the same TINKER program. For example, the following command line:

> **newton molecule -k test a a 0.01**

will invoke the NEWTON program on the structure file **molecule.xyz** using the keyfile **test.key**, automatic mode **[a]** for both the method and preconditioning, and **0.01** for the RMS gradient per atom termination criterion in kcal/mole/Å. Provided that the force field parameter set, *etc.* is provided in **test.key**, the above compuation will procede directly from the command line invocation without further interactive input.

USE ON WINDOWS 9X/NT SYSTEMS

TINKER executables for Microsoft PC systems should be run from the "DOS box" available via the MS-DOS Prompt program. The TINKER executable directory should be added to your path via the autoexec.bat file or similar. These DOS terminal windows are only able to scroll a small number of lines (!), so TINKER programs which generate large amounts of screen output should be run such that output will be redirected to a file. This can be accomplished by running the TINKER program in batch mode or by using the Unix-like output redirection build into DOS. For example, the command:

> **dynamic < molecule.inp > molecule.log**

will run the TINKER dynamic program taking input from the file **molecule.inp** and sending output to **molecule.log**. Also note that command line options as described above are available with the distributed TINKER executables.

If the distributed TINKER executables are directly from Windows by double clicking on the program icon, then the program will run in its own window. However, upon completion of the program the window will close and screen output will be lost. Any output files written by the program will, of course, still be available. The Windows behavior can be changed by adding the EXIT-PAUSE keyword to the keyfile.

USE ON MACINTOSH SYSTEMS

The TINKER executables can be run under MacOS by double clicking on a program icon. The program will run in its own window to which all "screen" output will be directed. Upon program termination the window will remain active pending a final return entered by the user which will close the window. Prior to the final return, the contents of the screen window can be saved to a file via the clipboard for permanent storage. Note that Macintosh uses ":" instead of "/" or "\" as the directory separator, so keyfiles transfered from other machines will need to be altered accordingly.

ATOM TYPES VS. ATOM CLASSES

Manipulation of atom "types" and the proliferation of parameters as atoms are further subdivided into new types is the bane of force field calculation. For example, if each

topologically distinct atom arising from the 20 natural amino acids is given a different atom type, then about 300 separate type are required (this ignores the "different" N- and C-terminal residues, diastereotopic hydrogens, *etc.*). However, these 300 types lead to literally thousands of different force field parameters. In fact, there are thousands of distinct torsional parameters alone. It is impossible at present to fully optimize each of these parameters; and even if we could, a great many of the parameters would be nearly identical. Two somewhat complimentary solutions are available to handle the proliferation of parameters. The first is to specify the molecular fragments to which a given parameter can be applied in terms of a chemical structure language, SMILES strings for example. Some commercial systems, such as the TRIPOS Sybyl software, make use of such a scheme to parse structures and assign force field parameters.

A second general approach is to use hierarchical cascades of parameter groups. TINKER uses a simple version of this scheme. Each TINKER force field atom has both an atom *type* number and an atom *class* number. The types are subsets of the atom classes, *i.e.*, several different atom types can belong to the same atom class. Force field parameters that are somewhat less sensitive to local environment, such as local geometry terms, are then provided and assigned based on atom class. Other energy parameters, such as electrostatic parameters, that are very environment dependent are assigned over the atom types. This greatly reduces the number of independent multiple-atom parameters like the four-atom torsional parameters.

CALCULATIONS ON PARTIAL STRUCTURES

Two methods are available for performing energetic calculations on portions or substructures within a full molecular system. TINKER allows division of the entire system into *active* and *inactive* parts which can be defined via keywords. In subsequent calculations, such as minimization or dynamics, only the active portions of the system are allowed to move. The force field engine responds to the active/inactive division by computing all energetic interactions involving at least one active atom; *i.e.*, any interaction whose energy can change with the motion of one or more active atoms is computed.

The second method for partial structure computation involves dividing the original system into a set of atom *groups*. As before, the groups can be specified via appropriate keywords. The current TINKER implementation allows specification of up to a maximum number of groups as given in the `sizes.i` dimensioning file. The groups must be disjoint in that no atom can belong to more than one group. Further keywords allow the user to specify which intra- and intergroup sets of energetic interactions will contribute to the total force field "energy". Weights for each set of interactions in the total energy can also be input. A specific energetic interaction is assigned to a particular intra- or intergroup set if all the atoms involved in the interaction belong to the group (intra-) or pair of groups (inter-). Interactions involving atoms from more than two groups are not computed.

Note that the groups method and active/inactive method use different assignment procedures for individual interactions. The active/inactive scheme is intended for situations where only a portion of a system is allowed to move, but the total energy needs to reflect the presence of the remaining inactive portion of the structure. The groups method is intended for use in rigid body motion and is needed for certain kinds of free energy perturbation calculations.

METAL COMPLEXES AND HYPERVALENT SPECIES

The distribution version of TINKER comes dimensioned for a maximum atomic coordination number of four as needed for standard organic compounds. In order to use TINKER for calculations on species containing higher coordination numbers, simply change the value of the parameter `maxval` in the master dimensioning file `sizes.i` and rebuilt the package. Note that this parameter value should not be set larger than necessary since large values can slow the execution of portions of some TINKER programs.

Many molecular mechanics approaches to inorganic and metal structures use an angle bending term which is softer than the usual harmonic bending potential. TINKER implements a Fourier bending term similar to that used by the Landis group's SHAPES force field. The parameters for specific Fourier angle terms are supplied via the ANGLEF parameter and keyword format. Note that a Fourier term will only be used for a particular angle if a corresponding harmonic angle term is not present in the parameter file.

NEIGHBOR METHODS FOR NONBONDED TERMS

In addition to standard double loop methods, the Method of Lights is available to speed neighbor searching. This method based on taking intersections of sorted atom lists can be much faster for problems where the cutoff distance is significantly smaller than half the maximal cell dimension. The current version of TINKER does not implement the "neighbor list" schemes common to many other simulation packages.

PERIODIC BOUNDARY CONDITIONS

Both spherical cutoff images or replicates of a cell are supported by all TINKER programs that implement periodic boundary conditions. Whenever the cutoff distance is too large for the minimum image to be the only relevant neighbor (*i.e.*, half the minimum box dimension for orthogonal cells), TINKER will automatically switch from the image formalism to use of replicated cells.

This version of TINKER contains an implementation of the particle mesh Ewald (PME) summation for partial charge electrostatics. PME is selected by default when force fields that use atomic partial charge models are coupled with periodic boundard conditions.

DISTANCE CUTOFFS FOR ENERGY FUNCTIONS

Polynomial energy switching over a window is used for terms whose energy is small near the cutoff distance. For monopole electrostatic interactions, which are quite large in typical cutoff ranges, a two polynomial multiplicative-additive shifted energy switch unique to TINKER is applied. The TINKER method is similar in spirit to the force switching methods of Steinbach and Brooks, *J. Comput. Chem.*, **15**, 667-683 (1994). While the particle mesh Ewald method is preferred when periodic boundary conditions are present, TINKER's shifted energy switch with reasonable switching windows is quite satisfactory for most routine modeling problems. The shifted energy switch minimizes the perturbation of the energy and the gradient at the cutoff to acceptable levels. Problems should arise only if the property you wish to monitor is known to require explicit inclusion of long range components (*i.e.*, calculation of the dielectric constant, *etc.*).

PARTICLE MESH EWALD SUMMATION

TINKER contains a version of the particle mesh Ewald summation technique for inclusion of long range electrostatics. The accuracy and speed of the PME calculation is

dependent on several interrelated parameters which control the Ewald coefficient, B-spline order, real-space cutoff distance and charge grid dimensions. By default TINKER will select a set of parameters which provide a reasonable compromise between accuracy and speed for most uses. Complete control over all the PME control parameters is available via the TINKER keyfile mechanism.

DISTANCE GEOMETRY METRIZATION

A much improved and very fast random pairwise metrization scheme is available which allows good sampling during trial distance matrix generation without the usual structural anomalies and CPU constraints of other metrization procedures. An outline of the methodology and its application to NMR NOE-based structure refinement is described in the paper by Hodsdon, *et al.* in *J. Mol. Biol.*, **264**, 585-602 (1996). We have obtained good results with something like the keyword phrase `trial-distribution pairwise 5`, which performs 5% partial random pairwise metrization. For structures over several hundred atoms, a value less than 5 for the percentage of metrization should be fine.

POLARIZABLE MULTIPOLE ELECTROSTATICS

Atomic multipole electrostatics through the quadrupole moment is supported by the current version of TINKER, as is either mutual or direct dipole polarization. All calculations are implemented via the Applequist-Dykstra Cartesian polytensor method and analytical energy and Cartesian gradient code is provided.

POTENTIAL ENERGY SMOOTHING

Versions of our Potential Smoothing and Search (PSS) methodology have been implemented within TINKER. This methods belong to the same general family as Scheraga's Diffusion Equation Method, Straub's Gaussian Density Annealing and Shalloway's Packet Annealing, but our algorithms reflect our own ongoing research in this area. In many ways the TINKER potential smoothing methods are the deterministic analog of stochastic simulated annealing. The PSS algorithms are very powerful, but are relatively new and are still undergoing modification, testing and calibration in the Ponder group. This version of TINKER also includes a basin-hopping conformational scanning algorithm in the program SCAN which is particularly effective on smoothed potential surfaces.

9. Descriptions of TINKER Routines

The distribution version of the TINKER package contains over 500 separate programs, subroutines and functions. This section contains a brief description of the purpose of most of these code units. Further information can be found in the comments located at the top of each source code file.

ACTIVE Subroutine

"active" sets the list of atoms that are used during each potential energy function calculation

ADDBOND Subroutine

"addbond" adds entries to the attached atoms list in order to generate a direct connection between two atoms

ADDSIDE Subroutine

"addside" builds the Cartesian coordinates file information for a single amino acid sidechain; coordinates are read from the Protein Data Bank file or found from internal coordinates; atom types are assigned and connectivity data generated

ADJACENT Function

"adjacent" finds an atom connected to atom "i1" other than atom "i2"; if no such atom exists, then the closest atom in space is returned

ALCHEMY Program

"alchemy" computes the free energy difference corresponding to a small perturbation by Boltzmann weighting the potential energy difference over a number of sample states; current version (incorrectly) considers the charge energy to be intermolecular in finding the perturbation energies

ANALYSIS Subroutine

"analysis" calls the series of routines needed to calculate the potential energy and perform energy partitioning analysis in terms of type of interaction and/or atom number

ANALYZE Program

"analyze" computes and displays the total potential; options are provided to partition the energy by atom or by potential function type; parameters used in computing interactions can also be displayed by atom; output of large energy interactions and the total electric charge and dipole moment are available

ANGLES Subroutine

"angles" finds the total number of bond angles and stores the atom numbers of the atoms defining each angle; for each angle to a tricoordinate central atom, the third bonded atom is stored for use in out-of-plane bending

ANNEAL Program

"anneal" performs a simulated annealing protocol by means of variable temperature molecular dynamics using either linear, exponential or sigmoidal cooling schedules

ANORM Function

"anorm" finds the norm (length) of a vector; used as a service routine by the Connolly surface area and volume computation

ARCHIVE Program

"archive" is a utility program for coordinate files which concatenates multiple coordinate sets into a single archive file, or extracts individual coordinate sets from an archive

ASET Subroutine

ATTACH Subroutine

"attach" sets up the lists of 1-3 and 1-4 connectivities starting from the previously determined list of attached atoms (ie, 1-2 connectivity)

BASEFILE Subroutine

"basefile" extracts from an input filename the portion consisting of any directory name and the base filename

BEEMAN Subroutine

"beeman" performs a single molecular dynamics time step by means of a Beeman multistep recursion formula; the actual coefficients are Brooks' "Better Beeman" values

BETACF Function

"betacf" computes a rapidly convergent continued fraction needed by routine "betai" to evaluate the cumulative Beta distribution

BETAI Function

"betai" evaluates the cumulative Beta distribution function as the probability that a random variable from a distribution with Beta parameters "a" and "b" will be less than "x"

BMAX Function

BNDANGLE Function

"bndangle" finds the value of the bond angle defined by three input atoms

BNDERR Function

"bnderr" is the distance bound error function and derivatives; this version implements the original and Havel's normalized lower bound penalty, the normalized version is preferred when lower bounds are small (as with NMR NOE restraints), the original penalty should be used if large lower bounds are present

BNDLENG Function

"bndleng" finds the value of the bond length defined by two input atoms

BONDS Subroutine

"bonds" finds the total number of covalent bonds and stores the atom numbers of the atoms defining each bond

BORN Subroutine

"born" computes the Born radius of each atom for use with the Macromodel GB/SA solvation model

BOUNDS Subroutine

"bounds" finds the center of mass of each molecule, translates any stray molecules back into the periodic box, and saves the offset of each atom relative to the molecular center of mass

BSET Subroutine

BSPLINE Subroutine

"bspline" calculates the coefficients and derivative coefficients for an n-th order B-spline approximation

BSSTEP Subroutine

CALENDAR Subroutine

"calendar" returns the current time as a set of integer values representing the year, month, day, hour, minute and second; only one of the various machine implementations included should be activated by removing comment characters

CELLATOM Subroutine

"cellatom" completes the addition of a symmetry related atom to a unit cell by updating the atom type and attachment arrays

CENTER Subroutine

"center" moves the weighted centroid of each coordinate set to the origin during least squares superposition

CFFTB Subroutine

"cfftb" computes the backward complex discrete Fourier transform, the Fourier synthesis

CFFTB1 Subroutine

CFFTF Subroutine

"cfftf" computes the forward complex discrete Fourier transform, the Fourier analysis

CFFTF1 Subroutine

CFFTI Subroutine

"cffti" initializes the array "wsave" which is used in both forward and backward transforms; the prime factorization of "n" together with a tabulation of the trigonometric functions are computed and stored in "wsave"

CFFTI1 Subroutine

CHIRER Function

"chirer" computes the chirality error and its derivatives with respect to atomic Cartesian coordinates as a sum the squares of deviations of chiral volumes from target values

CHIRIN Subroutine

"chirin" determines the target value for each chirality and planarity restraint as the signed volume of the parallelpiped spanned by vectors from a common atom to each of three other atoms

CHKSIZE Subroutine

"chksize" computes a measure of overall global structural expansion or compaction from the number of excess upper or lower bounds matrix violations

CHKTREE Subroutine

"chktree" tests a minimum energy structure to see if it belongs to the correct progenitor in the existing map

CHOLESKY Subroutine

"cholesky" uses a modified Cholesky method to solve the linear system Ax = b, returning "x" in "b"; "A" is assumed to be a real symmetric positive definite matrix with diagonal and upper triangle stored by rows in "A"; thus the actual size of the passed portion of "A" is nvar*(nvar+1)/2

CIRPLN Subroutine

CJKM Function

CLIMBER Subroutine

CLIMBRGD Subroutine

CLIMBROT Subroutine

CLIMBROT Subroutine

CLIMBXYZ Subroutine

CLOCK Subroutine

"clock" determines elapsed CPU time in seconds since the start of the job; only one of the implementations should be activated by removing comment characters from the code

CLUSTER Subroutine

"cluster" gets the partitioning of the system into groups and stores a list of the group to which each atom belongs

COLUMN Subroutine

"column" takes the off-diagonal Hessian elements stored as sparse rows and sets up indices to allow column access

COMMAND Subroutine

"command" uses the standard Unix-like iargc/getarg routines to get the number and values of arguments specified on the command line at program runtime

COMPRESS Subroutine

"compress" transfers only the non-buried tori from the temporary tori arrays to the final tori arrays

CONNECT Subroutine

"connect" sets up the attached atom arrays starting from a set of internal coordinates

CONNOLLY Subroutine

"connolly" uses the algorithms from the AMS/VAM programs of Michael Connolly to compute the analytical molecular surface area and volume of a collection of spherical atoms; thus it implements Fred Richards' molecular surface definition as a set of analytically defined spherical and toroidal polygons

CONTACT Subroutine

"contact" constructs the contact surface, cycles and convex faces

CONTROL Subroutine

"control" gets initial values for parameters that determine the output style and information level provided by TINKER

COORDS Subroutine

"coords" converts the three principal eigenvalues/vectors from the metric matrix into atomic coordinates, and calls a routine to compute the rms deviation from the bounds

CORRELATE Program

"correlate" computes the time correlation function of some property from the individual snapshot coordinate files taken from a molecular dynamics or other trajectory

CRYSTAL Program

"crystal" is a utility program which converts between fractional and Cartesian coordinates, and can generate full unit cells from asymmetric units

CUTOFFS Subroutine

"cutoffs" initializes and stores spherical energy cutoff distance windows, Hessian element and Ewald sum cutoffs, and the pairwise neighbor generation method

D1D2 Function

DELETE Subroutine

"delete" removes a specified atom from the Cartesian coordinates list and shifts the remaining atoms

DEPTH Function

DFTMOD Subroutine

"dftmod" computes the modulus of the discrete fourier transform of "bsarray", storing it into "bsmod"

DIAGQ Subroutine

"diagq" is a matrix diagonalization routine which is a conglomeration of the classical given, housec, and eigen algorithms with several modifications to increase the efficiency and accuracy

DIFFEQ Subroutine

"diffeq" performs the numerical integration of an ordinary differential equation using an adaptive stepsize method to solve the corresponding coupled first-order equations of the general form $dy_i/dx = f(x, y_1, ..., y_n)$ for $y_i = y_1, ..., y_n$

DIHEDRAL Function

"dihedral" finds the value of the dihedral angle in the range from -180 to +180 degrees defined by four input atoms

DIST2 Function

"dist2" finds the distance squared between two points; used as a service routine by the Connolly surface area and volume computation

DISTGEOM Program

"distgeom" uses a metric matrix distance geometry procedure to generate structures with interpoint distances that lie within specified bounds, with chiral centers that maintain chirality, and with torsional angles restrained to desired values; the user also has the ability to interactively inspect and alter the triangle smoothed bounds matrix prior to embedding

DMDUMP Subroutine

"dmdump" puts the distance matrix of the final structure into the upper half of a matrix, the distance of each atom to the centroid on the diagonal, and the individual terms of the bounds errors into the lower half of the matrix

DOCUMENT Program

"document" generates a formatted description of all the code modules or common blocks, a listing of all valid keywords, a list of include file dependencies as needed by a Unix-style Makefile, or a formatted force field parameter set summary

DOT Function

"dot" finds the dot product of two vectors

DROTMAT Subroutine

"drotmat" finds the derivative rotation matrices that convert multipoles from the local coordinate system to the global system

DROTMAT1 Subroutine

"drotmat1" finds the multipole rotation matrix derivatives for local coordinates defined via the "Z-then-X" method

DROTMAT2 Subroutine

"drotmat2" finds the multipole rotation matrix derivatives for local coordinates defined via the "Bisector" method

DROTPOLE Subroutine

"drotpole" computes the derivatives of the atomic multipoles in the global coordinate frame with respect to motion of the sites that define the local coordinate frame

DSTMAT Subroutine

"dstmat" selects a distance matrix containing values between the previously smoothed upper and lower bounds; the distance values are chosen from uniform distributions, in a triangle correlated fashion, or using random partial metrization

DYNAMIC Program

"dynamic" computes a molecular dynamics trajectory in any of several statistical mechanical ensembles with optional periodic boundaries and optional coupling to temperature and pressure baths; alternatively a stochastic dynamics trajectory can be generated

EANGANG Subroutine

"eangang" calculates the angle-angle potential energy

EANGANG1 Subroutine

"eangang1" calculates the angle-angle potential energy and first derivatives with respect to Cartesian coordinates

EANGANG2 Subroutine

"eangang2" calculates the angle-angle potential energy second derivatives with respect to Cartesian coordinates using finite difference methods

EANGANG2B Subroutine

"eangang2b" calculates the angle-angle first derivatives for a single interaction with respect to Cartesian coordinates; used in computation of finite difference second derivatives

EANGANG3 Subroutine

"eangang3" calculates the angle-angle potential energy; also partitions the energy among the atoms

EANGLE Subroutine

"eangle" calculates the angle bending potential energy; projected in-plane angles at trigonal centers or Fourier angle bending terms are optionally used

EANGLE1 Subroutine

"eangle1" calculates the angle bending potential energy and the first derivatives with respect to Cartesian coordinates; projected in-plane angles at trigonal centers or Fourier angle bending terms are optionally used

EANGLE2 Subroutine

"eangle2" calculates second derivatives of the angle bending energy for a single atom using a mixture of analytical and finite difference methods; projected in-plane angles at trigonal centers or Fourier angle bending terms are optionally used

EANGLE2A Subroutine

"eangle2a" calculates bond angle bending potential energy second derivatives with respect to Cartesian coordinates

EANGLE2B Subroutine

"eangle2b" computes projected in-plane bending first derivatives for a single angle with respect to Cartesian coordinates; used in computation of finite difference second derivatives

EANGLE3 Subroutine

"eangle3" calculates the angle bending potential energy, also partitions the energy among the atoms; projected in-plane angles at trigonal centers or Fourier angle bending terms are optionally used

EBOND Subroutine

"ebond" calculates the bond stretching energy

EBOND1 Subroutine

"ebond1" calculates the bond stretching energy and first derivatives with respect to Cartesian coordinates

EBOND2 Subroutine

"ebond2" calculates second derivatives of the bond stretching energy for a single atom at a time

EBOND3 Subroutine

"ebond3" calculates the bond stretching energy; also partitions the energy among the atoms

EBUCK Subroutine

"ebuck" calculates the van der Waals interaction energy using the Buckingham exp-6 formula

EBUCK1 Subroutine

"ebuck1" calculates the van der Waals energy and its first derivatives with respect to Cartesian coordinates using the Buckingham exp-6 formula

EBUCK2 Subroutine

"ebuck2" calculates the van der Waals second derivatives for a single atom at a time using the Buckingham exp-6 formula

EBUCK3 Subroutine

"ebuck3" calculates the van der Waals interaction energy using the Buckingham exp-6 formula and also partitions the energy among the atoms

EBUCK4 Subroutine

"ebuck4" calculates the Buckingham van der Waals interaction energy using the method of lights to locate neighboring atoms

EBUCK5 Subroutine

"ebuck5" calculates the Buckingham van der Waals interaction energy and its first derivatives using the method of lights to locate neighboring atoms

ECHARGE Subroutine

"echarge" calculates the charge-charge interaction energy

ECHARGE1 Subroutine

"echarge1" calculates the charge-charge interaction energy and first derivatives with respect to Cartesian coordinates

ECHARGE2 Subroutine

"echarge2" calculates second derivatives of the charge-charge interaction energy for a single atom

ECHARGE3 Subroutine

"echarge3" calculates the charge-charge interaction energy; also partitions the energy among the atoms

ECHARGE4 Subroutine

"echarge4" calculates the charge-charge interaction energy using the method of lights to locate neighboring atoms

ECHARGE5 Subroutine

"echarge5" calculates the charge-charge interaction energy and first derivatives with respect to Cartesian coordinates using the method of lights to locate neighboring atoms

ECHARGE6 Subroutine

"echarge6" calculates the smoothed charge-charge interaction energy

ECHARGE7 Subroutine

"echarge7" calculates the smoothed charge-charge interaction energy and first derivatives with respect to Cartesian coordinates

ECHARGE8 Subroutine

"echarge8" calculates second derivatives of the smoothed charge-charge interaction energy for a single atom

ECHARGE9 Subroutine

"echarge9" calculates the smoothed charge-charge interaction energy; also partitions the energy among the atoms

ECHGDPL Subroutine

"echgdpl" calculates the charge-dipole interaction energy

ECHGDPL1 Subroutine

"echgdpl1" calculates the charge-dipole interaction energy and first derivatives with respect to Cartesian coordinates

ECHGDPL2 Subroutine

"echgdpl2" calculates second derivatives of the charge-dipole interaction energy for a single atom

ECHGDPL3 Subroutine

"echgdpl3" calculates the charge-dipole interaction energy; also partitions the energy among the atoms

EDIPOLE Subroutine

"edipole" calculates the dipole-dipole interaction energy

EDIPOLE1 Subroutine

"edipole1" calculates the dipole-dipole interaction energy and first derivatives with respect to Cartesian coordinates

EDIPOLE2 Subroutine

"edipole2" calculates second derivatives of the dipole-dipole interaction energy for a single atom

EDIPOLE3 Subroutine

"edipole3" calculates the dipole-dipole interaction energy; also partitions the energy among the atoms

EGAUSS Subroutine

"egauss" calculates the van der Waals interaction energy using a Gaussian expansion approximation

EGAUSS1 Subroutine

"egauss1" calculates the van der Waals interaction energy and its first derivatives with respect to Cartesian coordinates using a Gaussian expansion approximation

EGAUSS2 Subroutine

"egauss2" calculates the van der Waals second derivatives for a single atom at a time using a Gaussian approximation

EGAUSS3 Subroutine

"egauss3" calculates the van der Waals interaction energy using a Gaussian approximation and also partitions the energy among the atoms

EGBSA Subroutine

"egbsa" calculates the generalized Born energy term for the Macromodel GB/SA solvation model

EGBSA1 Subroutine

"egbsa1" calculates the generalized Born energy and first derivatives with respect to Cartesian coordinates for the Macromodel GB/SA solvation model

EGBSA2 Subroutine

"egbsa2" calculates second derivatives of the generalized Born energy term for the Macromodel GB/SA solvation model

EGBSA3 Subroutine

"egbsa3" calculates the generalized Born energy term for the Macromodel GB/SA solvation model; also partitions the energy among the atoms

EGEOM Subroutine

"egeom" calculates the energy due to restraints on atomic positions, interatomic distances, dihedral angles, and Gaussian weighted molecular compactness

EGEOM1 Subroutine

"egeom1" calculates the potential energy and first derivatives with respect to Cartesian coordinates for restraints on atom positions, interatomic distances and dihedral angles

EGEOM2 Subroutine

"egeom2" calculates second derivatives of any restraints on atom positions, interatomic distances and dihedral angles

EGEOM3 Subroutine

"egeom3" calculates the energy due to restraints on atomic positions, interatomic distances, dihedral angles and Gaussian weighted molecular size; also partitions energy among the atoms

EHAL Subroutine

"ehal" calculates the van der Waals interaction energy using Halgren's Buffered 14-7 formula

EHAL1 Subroutine

"ehal1" calculates the van der Waals interaction energy and its first derivatives with respect to Cartesian coordinates using Halgren's Buffered 14-7 formula

EHAL2 Subroutine

"ehal2" calculates the van der Waals second derivatives for a single atom at a time using Halgren's Buffered 14-7 formula

EHAL3 Subroutine

"ehal3" calculates the van der Waals interaction energy using Halgren's Buffered 14-7 formula and also partitions the energy among the atoms

EHAL4 Subroutine

"ehal4" calculates the van der Waals interaction energy using Halgren's Buffered 14-7 formula and the method of lights to locate neighboring atoms

EHAL5 Subroutine

"ehal5" calculates the van der Waals interaction energy and its first derivatives using Halgren's Buffered 14-7 formula and the method of lights to locate neighboring atoms

EIGEN Subroutine

"eigen" uses the power method to compute the largest eigenvalues and eigenvectors of the metric matrix, "valid" is set true if the first three eigenvalues are positive

EIGENRGD Subroutine

EIGENROT Subroutine

EIGENROT Subroutine

EIGENROT Subroutine

EIGENXYZ Subroutine

EIMPROP Subroutine

"eimprop" calculates the improper dihedral potential energy

EIMPROP1 Subroutine

"eimprop1" calculates improper dihedral energy and its first derivatives with respect to Cartesian coordinates

EIMPROP2 Subroutine

"eimprop2" calculates second derivatives of the improper dihedral angle energy for a single atom

EIMPROP3 Subroutine

"eimprop3" calculates the improper dihedral potential energy; also partitions the energy terms among the atoms

EIMPTOR Subroutine

"eimptor" calculates the improper torsion potential energy

EIMPTOR1 Subroutine

"eimptor1" calculates improper torsion energy and its first derivatives with respect to Cartesian coordinates

EIMPTOR2 Subroutine

"eimptor2" calculates second derivatives of the improper torsion energy for a single atom

EIMPTOR3 Subroutine

"eimptor3" calculates the improper torsion potential energy; also partitions the energy terms among the atoms

ELJ Subroutine

"elj" calculates the van der Waals interaction energy using the Lennard-Jones 6-12 formalism

ELJ1 Subroutine

"elj1" calculates the van der Waals energy and its first derivatives with respect to Cartesian coordinates using the Lennard-Jones 6-12 formalism

ELJ2 Subroutine

"elj2" calculates the van der Waals second derivatives for a single atom at a time using the Lennard-Jones 6-12 formalism

ELJ3 Subroutine

"elj3" calculates the van der Waals interaction energy using the Lennard-Jones 6-12 formalism and also partitions the energy among the atoms

ELJ4 Subroutine

"elj4" calculates the Lennard-Jones van der Waals interaction energy using the method of lights to locate neighboring atoms

ELJ5 Subroutine

"elj5" calculates the Lennard-Jones van der Waals interaction energy and its first derivatives using the method of lights to locate neighboring atoms

EMBED Subroutine

"embed" is a distance geometry routine patterned after the ideas of Gordon Crippen, Irwin Kuntz and Tim Havel; it takes as input a set of upper and lower bounds on the interpoint distances, chirality restraints and torsional restraints, and attempts to generate a set of coordinates that satisfy the input bounds and restraints

EMM3HB Subroutine

"emm3hb" calculates the van der Waals and hydrogen bonding interaction energy using the MM3 exp-6 formula with directional charge transfer hydrogen bonding

EMM3HB1 Subroutine

"emm3hb1" calculates the van der Waals and hydrogen bonding energy and its first derivatives with respect to Cartesian coordinates using the MM3 exp-6 formula with directional charge transfer hydrogen bonding

EMM3HB2 Subroutine

"emm3hb2" calculates the van der Waals and hydrogen bonding second derivatives for a single atom at a time using the MM3 exp-6 formula with directional charge transfer hydrogen bonding

EMM3HB3 Subroutine

"emm3hb3" calculates the van der Waals and hydrogen bonding interaction energy using MM3 exp-6 formula with directional charge transfer hydrogen bonding; also partitions the energy among the atoms

EMM3HB4 Subroutine

"emm3hb4" calculates the MM3 exp-6 van der Waals and hydrogen bonding interaction energy using the method of lights to locate neighboring atoms

EMM3HB5 Subroutine

"emm3hb5" calculates the MM3 exp-6 van der Waals and hydrogen bonding interaction energy and its first derivatives using the method of lights to locate neighboring atoms

EMPIK Subroutine

"empik" computes the permanent multipole and induced dipole energies between a specified pair of atomic multipole sites

EMPIK1 Subroutine

"empik1" computes the permanent multipole and induced dipole energies and derivatives between a pair of multipole sites

EMPOLE Subroutine

"empole" calculates the electrostatic energy due to atomic multipole interactions and dipole polarizability

EMPOLE1 Subroutine

"empole1" calculates the multipole and dipole polarization energy and derivatives with respect to Cartesian coordinates

EMPOLE2 Subroutine

"empole2" calculates second derivatives of the multipole and dipole polarization energy for a single atom at a time

EMPOLE2B Subroutine

"empole2b" computes multipole and dipole polarization first derivatives for a single angle with respect to Cartesian coordinates; used to get finite difference second derivatives

EMPOLE3 Subroutine

"empole3" calculates the electrostatic energy due to atomic multipole interactions and dipole polarizability, and also partitions the energy among the atoms

ENERGY Function

"energy" calls the subroutines to calculate the potential energy terms and sums up to form the total energy

EOPBEND Subroutine

"eopbend" computes the out-of-plane bend potential energy at trigonal centers as per Allinger's MM2 and MM3 forcefields

EOPBEND1 Subroutine

"eopbend1" computes the out-of-plane bend potential energy and first derivatives at trigonal centers as per Allinger's MM2 and MM3 forcefields

EOPBEND2 Subroutine

"eopbend2" calculates second derivatives of the out-of-plane bend energy as per Allinger's MM2 and MM3 forcefields for a single atom using finite difference methods

EOPBEND2B Subroutine

"eopbend2b" calculates out-of-plane bending first derivatives for a single angle with respect to Cartesian coordinates; used in computation of finite difference second derivatives

EOPBEND3 Subroutine

"eopbend3" computes the out-of-plane bend potential energy at trigonal centers as per Allinger's MM2 and MM3 forcefields; also partitions the energy among the atoms

EPME Subroutine

"epme" computes the reciprocal space energy for a particle mesh Ewald summation over partial charges

EPME1 Subroutine

"epme1" computes the reciprocal space energy and first derivatives for a particle mesh Ewald summation

EPUCLC Subroutine

ERF Function

"erf" computes a numerical approximation to the value of the error function via a Chebyshev approximation

ERFC Function

"erfc" computes a numerical approximation to the value of the error function complement via a Chebyshev approximation

ERFCORE Subroutine

"erfcore" evaluates erf(x) or erfc(x) for a real argument x; this routine is called as the numerical kernel by two simple functions: "erf" and "erfc"; if "calerf" is called with a mode value of 0 it returns erf, a mode of 1 returns erfc; it uses rational functions that theoretically approximate erf(x) and erfc(x) to at least 18 significant decimal digits

ERFIK Subroutine

ERFINV Function

"erfinv" evaluates the inverse of the error function erf for a real argument in the range (-1,1) using a rational function approximation followed by cycles of Newton-Raphson correction

ERROR Subroutine

"error" is the error handling routine for the Connolly surface area and volume computation

ERXNFLD Subroutine

"erxnfld" calculates the macroscopic reaction field energy

ERXNFLD1 Subroutine

"erxnfld1" calculates the macroscopic reaction field energy and derivatives with respect to Cartesian coordinates

ERXNFLD2 Subroutine

"erxnfld2" calculates second derivatives of the macroscopic reaction field energy for a single atom at a time

ERXNFLD3 Subroutine

"erxnfld3" calculates the macroscopic reaction field energy, and also partitions the energy among the atoms

ESOLV Subroutine

"esolv" calculates the macroscopic solvation energy via either the Eisenberg-McLachlan ASP, Ooi-Scheraga SASA or Macromodel GB/SA solvation model

ESOLV1 Subroutine

"esolv1" calculates the macroscopic solvation energy and first derivatives with respect to Cartesian coordinates using either the Eisenberg-McLachlan ASP, Ooi-Scheraga SASA or Macromodel GB/SA solvation model

ESOLV2 Subroutine

"esolv2" calculates second derivatives of the macroscopic solvation energy using either the Eisenberg-McLachlan ASP, Ooi-Scheraga SASA or Macromodel GB/SA solvation model

ESOLV3 Subroutine

"esolv3" calculates the macroscopic solvation energy using either the Eisenberg-McLachlan ASP, Ooi-Scheraga SASA or Macromodel GB/SA solvation model; also partitions the energy among the atoms

ESTRBND Subroutine

"estrbnd" calculates the stretch-bend potential energy

ESTRBND1 Subroutine

"estrbnd1" calculates the stretch-bend potential energy and first derivatives with respect to Cartesian coordinates

ESTRBND2 Subroutine

"estrbnd2" calculates the stretch-bend potential energy second derivatives with respect to Cartesian coordinates

ESTRBND3 Subroutine

"estrbnd3" calculates the stretch-bend potential energy; also partitions the energy among the atoms

ESTRTOR Subroutine

"estrtor" calculates the stretch-torsion potential energy

ESTRTOR1 Subroutine

"estrtor1" calculates the stretch-torsion energy and first derivatives with respect to Cartesian coordinates

ESTRTOR2 Subroutine

"estrtor2" calculates the stretch-torsion potential energy second derivatives with respect to Cartesian coordinates

ESTRTOR3 Subroutine

"estrtor3" calculates the stretch-torsion potential energy; also partitions the energy terms among the atoms

ETORS Subroutine

"etors" calculates the torsional potential energy

ETORS1 Subroutine

"etors1" calculates torsional potential energy and first derivatives with respect to Cartesian coordinates

ETORS2 Subroutine

"etors2" calculates second derivatives of the torsional energy for a single atom

ETORS3 Subroutine

"etors3" calculates the torsional potential energy; also partitions the energy terms among the atoms

EUREY Subroutine

"eurey" calculates the Urey-Bradley 1-3 interaction energy

EUREY1 Subroutine

"eurey1" calculates the Urey-Bradley interaction energy and its first derivatives with respect to Cartesian coordinates

EUREY2 Subroutine

"eurey2" calculates second derivatives of the Urey-Bradley interaction energy for a single atom at a time

EUREY3 Subroutine

"eurey3" calculates the Urey-Bradley energy; also partitions the energy among the atoms

EWALD Subroutine

"ewald" calculates the Ewald summation interaction energy

EWALD1 Subroutine

"ewald1" calculates the Ewald summation interaction energy and first derivatives with respect to Cartesian coordinates

EWALD2 Subroutine

"ewald2" calculates second derivatives of the Ewald summation interaction energy for a single atom

EWALD3 Subroutine

"ewald3" calculates the Ewald summation interaction energy; also partitions the energy among the atoms

EWALDCOF Subroutine

EXPLORE Subroutine

"explore" uses simulated annealing on an initial crude embedded distance geoemtry structure to refine versus the bound, chirality, planarity and torsional error functions

EXTRA Subroutine

"extra" calculates any additional user defined potential energy contribution

EXTRA1 Subroutine

"extra1" calculates any additional user defined potential energy contribution and its first derivatives

EXTRA2 Subroutine

"extra2" calculates second derivatives of any additional user defined potential energy contribution for a single atom at a time

EXTRA3 Subroutine

"extra3" calculates any additional user defined potential contribution and also partitions the energy among the atoms

FATAL Subroutine

"fatal" terminates execution due to a user request, a severe error or some other nonstandard condition

FFTBACK Subroutine

FFTFRONT Subroutine

FFTSETUP Subroutine

FIELD Subroutine

"field" sets the force field potential energy functions from a parameter file and modifications specified in a keyfile

FINAL Subroutine

"final" performs any final program actions, prints a status message, and then pauses if necessary to avoid killing the execution window

FINDATM Subroutine

"findatm" locates a specific PDB atom name type within a range of atoms from the PDB file, returns zero if the name type was not found

FIXPDB Subroutine

"fixpdb" corrects problems with the AMBER and CHARMM/XPLOR PDB files by converting atom names to the PDB standards

FRACDIST Subroutine

"fracdist" computes a normalized distribution of the pairwise fractional distances between the smoothed upper and lower bounds

FREEUNIT Function

"freeunit" finds an unopened Fortran I/O unit and returns its numerical value from 1 to 99; the units already assigned to "input" and "iout" (usually 5 and 6) are skipped since they have special meaning as the default I/O units

GAMMLN Function

"gammln" uses a series expansion due to Lanczos to compute the natural logarithm of the Gamma function at "x" in [0,1]

GDA Program

"gda" implements Gaussian Density Annealing (GDA) algorithm for global optimization via simulated annealing

GDA1 Function

GDA2 Function

GDA3 Subroutine

GDASTAT Subroutine

GENDOT Subroutine

"gendot" finds the coordinates of a specified number of surface points for a sphere with the input radius and coordinate center

GEODESIC Subroutine

"geodesic" smooths the upper and lower distance bounds via the triangle inequality using a sparse matrix version of a shortest path algorithm

GETHYDRO Subroutine

"gethydro" finds the side chain hydrogen atoms for a single amino acid residue and copies the names and coordinates to the Protein Data Bank file

GETIME Subroutine

"getime" gets elapsed CPU time in seconds for an interval

GETINT Subroutine

"getint" asks for an internal coordinate file name, then reads the internal coordinates and computes Cartesian coordinates

GETKEY Subroutine

"getkey" finds a valid keyfile and stores its contents as line images for subsequent keyword parameter searching

GETMOL2 Subroutine

"getmol2" asks for a Sybyl MOL2 molecule file name, then reads the coordinates from the file

GETNUMB Subroutine

"getnumb" searchs an input string from left to right for an integer and puts the numeric value in "number"; returns zero with "next" unchanged if no integer value is found

GETPDB Subroutine

"getpdb" asks for a Protein Data Bank file name, then reads in the coordinates file

GETPRB Subroutine

"getprb" tests for a possible probe position at the interface between three neighboring atoms

GETPRM Subroutine

"getprm" finds the potential energy parameter file and then opens and reads the parameters

GETREF Subroutine

"getref" copies structure information from the reference area into the standard variables for the current system structure

GETSEQ Subroutine

"getseq" asks the user for the amino acid sequence and torsional angle values needed to define a peptide

GETSIDE Subroutine

"getside" finds the side chain heavy atoms for a single amino acid residue and copies the names and coordinates to the Protein Data Bank file

GETSTRING Subroutine

"getstring" searchs for a quoted text string within an input character string; the region between the first and second quotes is returned as the "text"; if the actual text is too long, only the first part is returned

GETTEXT Subroutine

"gettext" searchs an input string for the first string of non-blank characters; the region from a non-blank character to the first blank space is returned as "text"; if the actual text is too long, only the first part is returned

GETTOR Subroutine

"gettor" tests for a possible torus position at the interface between two atoms, and finds the torus radius, center and axis

GETWORD Subroutine

"getword" searchs an input string for the first alphabetic character (A-Z or a-z); the region from this first character to the first blank space or comma is returned as a "word"; if the actual word is too long, only the first part is returned

GETXYZ Subroutine

"getxyz" asks for a Cartesian coordinate file name, then reads in the coordinates file

GRADIENT Subroutine

"gradient" calls subroutines to calculate the potential energy and first derivatives with respect to Cartesian coordinates

GRADRGD Subroutine

"gradrgd" calls subroutines to calculate the potential energy and first derivatives with respect to rigid body coordinates

GRADROT Subroutine

"gradrot" calls subroutines to calculate the potential energy and its torsional first derivatives

GRAFIC Subroutine

"grafic" outputs the upper & lower triangles and diagonal of a square matrix in a schematic form for visual inspection

GROUPS Subroutine

"groups" tests a set of atoms to see if all are members of a single atom group or a pair of atom groups; if so, then the correct intra- or intergroup weight is assigned

GYRATE Subroutine

"gyrate" computes the radius of gyration of a molecular system from its atomic coordinates

HANGLE Subroutine

"hangle" constructs the hybrid angle bending parameters given an initial state, final state and "lambda" value

HATOM Subroutine

"hatom" assigns a new atom type to each hybrid site

HBOND Subroutine

"hbond" constructs the hybrid bonded interaction parameters given an initial state, final state and mutation parameter "lambda"

HCHARGE Subroutine

"hcharge" constructs the hybrid charge interaction energy parameters given an initial state, final state and "lambda"

HDIPOLE Subroutine

"hdipole" constructs the hybrid dipole interaction energy parameters given an initial state, final state and "lambda"

HESSIAN Subroutine

"hessian" calls subroutines to calculate the Hessian elements for each atom in turn with respect to Cartesian coordinates

HESSRGD Subroutine

"hessrgd" computes the numerical Hessian elements with respect to rigid body coordinates; either the full matrix or just the diagonal can be calculated; the full matrix needs 6*ngroup+1 gradient evaluations while the diagonal requires just two gradient calls

HESSROT Subroutine

"hessrot" computes the numerical Hessian elements with respect to torsional angles; either the full matrix or just the diagonal can be calculated; the full matrix needs nomega+1 gradient evaluations while the diagonal requires just two gradient calls

HIMPTOR Subroutine

"himptor" constructs the hybrid improper torsional parameters given an initial state, final state and mutation parameter "lambda"

HSTRBND Subroutine

"hstrbnd" constructs the hybrid stretch-bend parameters given an initial state, final state and "lambda" value

HSTRTOR Subroutine

"hstrtor" constructs the hybrid stretch-torsion parameters given an initial state, final state and "lambda" value

HTORS Subroutine

"htors" constructs the hybrid torsional parameters for a given initial state, final state, and "lambda" value

HVDW Subroutine

"hvdw" constructs the hybrid van der Waals interaction parameters given an initial state, final state and value of the mutation parameter "lambda"

HYBRID Subroutine

"hybrid" constructs the hybrid hamiltonian for a specified initial state, final state and mutation parameter "lambda"

IJK_PT Subroutine

IMAGE Subroutine

"image" takes the components of pairwise distance between two points in the same or neighboring periodic boxes and converts to the components of the minimum image distance

IMPOSE Subroutine

"impose" performs the least squares best superposition of two atomic coordinate sets via a quaternion method; upon return, the first coordinate set is unchanged while the second set is translated and rotated to give best fit; the final root mean square fit is returned in "rmsvalue"

INDUCE Subroutine

"induce" computes the induced dipole moment at each polarizable site due to direct or mutual polarization

INEDGE Subroutine

"inedge" inserts a concave edge into the linked list for its temporary torus

INERTIA Subroutine

"inertia" computes the principal moments of inertia for the system, and optionally translates the center of mass to the origin and rotates the principal axes onto the global axes

INITER Function

"initer" is the initial error function and derivatives for a distance geometry embedding; it includes components from the local geometry and torsional restraint errors

INITIAL Subroutine

"initial" sets up original values for some parameters and variables that might not otherwise get initialized

INITPRM Subroutine

"initprm" completely initializes a force field by setting all values to zero prior to reading in a parameter set

INITRES Subroutine

"initres" sets names for biopolymer residue types used in PDB file conversion and automated generation of structures

INITROT Subroutine

"initrot" asks for torsional angles which are to be rotated in subsequent computation, it will automatically locate all rotatable single bonds if desired; assumes that an appropriate internal coordinates file has already been read in

INSERT Subroutine

"insert" adds the specified atom to the Cartesian coordinates list and shifts the remaining atoms

INTEDIT Program

"intedit" allows the user to extract information from or alter the values within an internal coordinates file

INTXYZ Program

"intxyz" takes as input an internal coordinates file, converts to and then writes out Cartesian coordinates

INVBETA Function

"invbeta" computes the inverse Beta distribution function via a combination of Newton iteration and bisection search

INVERT Subroutine

"invert" inverts a matrix using the Gauss-Jordan method

IPEDGE Subroutine

"ipedge" inserts convex edge into linked list for atom

JACOBI Subroutine

"jacobi" performs a matrix diagonalization of a real symmetric matrix by the method of Jacobi rotations

KANGANG Subroutine

"kangang" assigns the parameters for angle-angle cross term interactions and processes new or changed parameter values

KANGLE Subroutine

"kangle" assigns the force constants and ideal angles for the bond angles; also processes new or changed parameters

KATOM Subroutine

"katom" assigns an atom type definitions to each atom in the structure and processes any new or changed values

KBOND Subroutine

"kbond" assigns a force constant and ideal bond length to each bond in the structure and processes any new or changed parameter values

KCHARGE Subroutine

"kcharge" assigns partial charges to the atoms within the structure and processes any new or changed values

KDIPOLE Subroutine

"kdipole" assigns bond dipoles to the bonds within the structure and processes any new or changed values

KEWALD Subroutine

"kewald" assigns particle mesh Ewald summation parameters for a periodic box and performs initialization

KIMPROP Subroutine

"kimprop" assigns potential parameters to each improper dihedral in the structure and processes any changed values

KIMPTOR Subroutine

"kimptor" assigns torsional parameters to each improper torsion in the structure and processes any changed values

KMPOLE Subroutine

"kmpole" assigns atomic multipole moments to the atoms of the structure and processes any new or changed values

KOPBEND Subroutine

"kopbend" assigns the force constants for out-of-plane bends; also processes any new or changed parameter values

KORBIT Subroutine

"korbit" assigns pi-orbital parameters to conjugated systems and processes any new or changed parameters

KPOLAR Subroutine

"kpolar" assigns atomic dipole polarizabilities to the atoms within the structure and processes any new or changed values

KSTRBND Subroutine

"kstrbnd" assigns the parameters for the stretch-bend interactions and processes new or changed parameter values

KSTRTOR Subroutine

"kstrtor" assigns stretch-torsion parameters to torsions needing them, and processes any new or changed values

KTORS Subroutine

"ktors" assigns torsional parameters to each torsion in the structure and processes any new or changed values

KUREY Subroutine

"kurey" assigns the force constants and ideal distances for the Urey-Bradley 1-3 interactions; also processes any new or changed parameter values

KVDW Subroutine

"kvdw" assigns the parameters to be used in computing the van der Waals interactions and processes any new or changed values for these parameters

LATTICE Subroutine

"lattice" stores the periodic box dimensions, finds angles to be used in computing fractional coordinates, and decides between images and replicates for the boundary conditions

LIGHTS Subroutine

"lights" computes the set of nearest neighbor interactions using the method of lights algorithm

LMQN Subroutine

"lmqn" is a generalized conjugate gradient optimization routine which implements steepest descent, Fletcher-Reeves CG, Polak-Ribiere CG, Hestenes-Stiefel CG, Powell-Beale CG with restarts, and a memoryless BFGS quasi-Newton method

LMSTEP Subroutine

"lmstep" computes the Levenberg-Marquardt step during a nonlinear least squares calculation; this version is based upon ideas from the Minpack routine LMPAR together with with the internal doubling strategy of Dennis and Schnabel

LOCALMIN Subroutine

"localmin" is used during normal mode local search to perform a Cartesian coordinate energy minimization

LOCALRGD Subroutine

"localrgd" is used during the PSS local search procedure to perform a rigid body energy minimization

LOCALROT Subroutine

"localrot" is used during the PSS local search procedure to perform a torsional space energy minimization

LOCERR Function

"locerr" is the local geometry error function and derivatives including the 1-2, 1-3 and 1-4 distance bound restraints

LOWCASE Subroutine

"lowcase" converts a text string to all lower case letters

MAJORIZE Subroutine

"majorize" refines the projected coordinates by attempting to minimize the least square residual between the trial distance matrix and the distances computed from the coordinates

MAKE27 Subroutine

"make27" replicates the coordinates of a single unit cell to give a full block of 27 (3x3x3 cube) unit cells

MAKEINT Subroutine

"makeint" converts Cartesian to internal coordinates where selection of internal coordinates is controlled by "mode"

MAKEPDB Subroutine

"makexyz" converts a set of Cartesian coordinates to Protein Data Bank format with special handling for systems consisting of polypeptide chains, ligands and water molecules

MAKEREF Subroutine

"makeref" copies the information contained in the "xyz" file of the current structure into corresponding reference areas

MAKEXYZ Subroutine

"makexyz" generates a complete set of Cartesian coordinates for a full structure from the internal coordinate values

MAPCHECK Subroutine

"mapcheck" checks the current minimum energy structure for possible addition to the master list of local minima

MAXWELL Function

"maxwell" returns a speed in Angstroms/picosecond randomly selected from a 3-D Maxwell-Boltzmann distribution for the specified particle mass and system temperature

MDINIT Subroutine

"mdinit" initializes the velocities and accelerations for a molecular dynamics trajectory, including restarts

MDREST Subroutine

"mdrest" computes and then removes any translational or rotational kinetic energy of the center of mass

MDSTAT Subroutine

"mdstat" is called at each molecular dynamics time step to form statistics on various average values and fluctuations, and to periodically save the state of the trajectory

MEASFN Subroutine

MEASFP Subroutine

MEASFS Subroutine

MEASPM Subroutine

"measpm" computes the volume of a single prism section of the full interior polyhedron

MECHANIC Subroutine

"mechanic" sets up needed parameters for the potential energy calculation and reads in many of the user selectable options

MERGE Subroutine

"merge" combines the reference and current structures into a single new "current" structure containing the reference atoms followed by the atoms of the current structure

METRIC Subroutine

"metric" takes as input the trial distance matrix and computes the metric matrix of all possible dot products between the atomic vectors and the center of mass using the law of cosines and a formula for the distances to the center of mass

MIDERR Function

"miderr" is the secondary error function and derivatives for a distance geometry embedding; it includes components from the distance bounds, local geometry, chirality and torsional restraint errors

MINIMIZ1 Function

"minimiz1" is a service routine that computes the energy and gradient for a nonlinear conjugate gradient optimization in Cartesian coordinate space

MINIMIZE Program

"minimize" performs an energy minimization in Cartesian coordinate space using a nonlinear conjugate gradient method

MINIROT Program

"minirot" performs an energy minimization in torsional angle space using a nonlinear conjugate gradient method

MINIROT1 Function

"minirot1" is a service routine that computes the energy and gradient for a nonlinear conjugate gradient optimization in torsional angle space

MINPATH Subroutine

"minpath" is a routine for finding the triangle smoothed upper and lower bounds of each atom to a specified root atom using a sparse variant of the Bellman-Ford shortest path algorithm

MMID Subroutine

"mmid" implements a modified midpoint method to advance the integration of a set of first order differential equations

MODECART Subroutine

MODESRCH Subroutine

MODETORS Subroutine

MODETORS Subroutine

MODULI Subroutine

"moduli" sets the moduli of the inverse discrete fourier transform of the B-splines; bsmod[1-3] hold these values, nfft[1-3] are the grid dimensions, order is the order of B-spline approximation

MOLECULE Subroutine

"molecule" counts the molecules, assigns each atom to its molecule and computes the mass of each molecule

MUTATE Subroutine

"mutate" constructs the hybrid hamiltonian for a specified initial state, final state and mutation parameter "lambda"

NEIGHBOR Subroutine

"neighbor" finds all of the neighbors of each atom

NEWATM Subroutine

"newatm" creates and defines an atom needed for the Cartesian coordinates file, but which may not present in the original Protein Data Bank file

NEWTON Program

"newton" performs an energy minimization in Cartesian coordinate space using a truncated Newton method

NEWTON1 Function

"newton1" is a service routine that computes the energy and gradient for truncated Newton optimization in Cartesian coordinate space

NEWTON2 Subroutine

"newton2" is a service routine that computes the sparse matrix Hessian elements for truncated Newton optimization in Cartesian coordinate space

NEWTROT Program

"newtrot" performs an energy minimization in torsional angle space using a truncated Newton conjugate gradient method

NEWTROT1 Function

"newtrot1" is a service routine that computes the energy and gradient for truncated Newton conjugate gradient optimization in torsional angle space

NEWTROT2 Subroutine

"newtrot2" is a service routine that computes the sparse matrix Hessian elements for truncated Newton optimization in torsional angle space

NEXTARG Subroutine

"nextarg" finds the next unused command line argument and returns it in the input character string

NEXTTEXT Function

"nexttext" finds and returns the location of the first non-blank character within an input text string; zero is returned if no such character is found

NORMAL Function

"normal" generates a random number from a normal Gaussian distribution with a mean of zero and a variance of one

NUMBER Function

"number" converts a text numeral into an integer value; the input string must contain only numeric characters

NUMERAL Subroutine

"numeral" converts an input integer number into the corresponding right- or left-justified text numeral

NUMGRAD Subroutine

"numgrad" computes the gradient of the objective function "fvalue" with respect to Cartesian coordinates of the atoms via a two-sided numerical differentiation

OCVM Subroutine

"ocvm" is an optimally conditioned variable metric nonlinear optimization routine without line searches

OLDATM Subroutine

"oldatm" get the Cartesian coordinates for an atom from the Protein Data Bank file, then assigns the atom type and atomic connectivities

OPTIMIZ1 Function

"optimiz1" is a service routine that computes the energy and gradient for optimally conditioned variable metric optimization in Cartesian coordinate space

OPTIMIZE Program

"optimize" performs energy minimization in Cartesian coordinate space using an optimally conditioned variable metric method

OPTIROT Program

"optirot" performs an energy minimization in torsional angle space using an optimally conditioned variable metric method

OPTIROT1 Function

"optirot1" is a service routine that computes the energy and gradient for optimally conditioned variable metric optimization in torsional angle space

OPTRIGID Program

"optrigid" performs an energy minimization of rigid body atom groups using an optimally conditioned variable metric method

OPTRIGID1 Function

"optrigid1" is a service routine that computes the energy and gradient for gradient optimization of rigid bodies

ORBITAL Subroutine

"orbital" finds and organizes lists of atoms in a pisystem, bonds connecting pisystem atoms and torsions whose two central atoms are both pisystem atoms

ORIENT Subroutine

"orient" computes a set of reference Cartesian coordinates in standard orientation for each rigid body atom group

ORTHOG Subroutine

"orthog" performs an orthogonalization of an input matrix via the modified Gram-Schmidt algorithm

OVERLAP Subroutine

"overlap" computes the overlap for two parallel p-orbitals given the atomic numbers and distance of separation

PASSB Subroutine

PASSB2 Subroutine

PASSB3 Subroutine

PASSB4 Subroutine

PASSB5 Subroutine

PASSF Subroutine

PASSF2 Subroutine

PASSF3 Subroutine

PASSF4 Subroutine

PASSF5 Subroutine

PATH Program

"path" locates a series of structures equally spaced along a conformational pathway connecting the input reactant and product structures; a series of constrained optimizations orthogonal to the path is done via Lagrangian multipliers

PATH1 Function

PATHPNT Subroutine

"pathpnt" finds a structure on the synchronous transit path with the specified path value "t"

PATHSCAN Subroutine

"pathscan" makes a scan of a synchronous transit pathway by computing structures and energies for specific path values

PATHVAL Subroutine

"pathval" computes the synchronous transit path value for the specified structure

PDBATM Subroutine

"pdbatm" adds an atom to the Protein Data Bank file

PDBXYZ Program

"pdbxyz" takes as input a Protein Data Bank file and then converts to and writes out a Cartesian coordinates file and, for polypeptides, a sequence file

PIALTER Subroutine

"pialter" first modifies bond lengths and force constants according to the standard bond slope parameters and the bond order values stored in "pnpl"; also alters some 2-fold torsional parameters based on the bond-order * beta matrix

PIMOVE Subroutine

"pimove" rotates the vector between atoms "list(1)" and "list(2)" so that atom 1 is at the origin and atom 2 along the x-axis; the atoms defining the respective planes are also moved and their bond lengths normalized

PIPLANE Subroutine

"piplane" selects the three atoms which specify the plane perpendicular to each p-orbital; the current version will fail in certain situations, including ketenes, allenes, and isolated or adjacent triple bonds

PISCF Subroutine

"piscf" performs an scf molecular orbital calculation for the pisystem using a modified Pariser-Parr-Pople method

PITILT Subroutine

"pitilt" calculates for each pibond the ratio of the actual p-orbital overlap integral to the ideal overlap if the same orbitals were perfectly parallel

PLACE Subroutine

"place" finds the probe sites by putting the probe sphere tangent to each triple of neighboring atoms

POLYP Subroutine

POTNRG Function

POTOFF Subroutine

"potoff" clears the forcefield definition by turning off the use of each of the potential energy functions

POWER Subroutine

"power" uses the power method with deflation to compute the few largest eigenvalues and eigenvectors of a symmetric matrix

PRECISE Function

"precise" finds one of three machine precision values

PRECOND Subroutine

"precond" solves a simplified version of the Newton equations Ms = r, and uses the result to precondition linear conjugate gradient iterations on the full Newton equations in "tnsolve"

PRESSURE Subroutine

"pressure" uses the internal virial to find the pressure in a periodic box and maintains a constant desired pressure by scaling the coordinates via coupling to an external constant pressure bath

PRMKEY Subroutine

"field" parses a text string to extract keywords related to force field potential energy functional forms and constants

PROJCT Subroutine

PROMO Subroutine

"promo" writes a short message containing info about the TINKER program package and the copyright notice

PROPERTY Function

"property" takes two input coordinate sets and computes the value of the property for which the time correlation function is being accumulated

PROTEIN Program

"protein" builds the internal and Cartesian coordinates of a polypeptide from the amino acid sequence and torsional angle values for the peptide backbone and side chains

PROTEUS Subroutine

"proteus" builds up the internal coordinates for an amino acid sequence from the phi, psi, omega and chi values

PRTBIOS Subroutine

"prtbios" writes out a set of Cartesian coordinates for all active atoms in the Biosym Insight II archive format

PRTDYN Subroutine

"prtdyn" writes out the information needed to restart a molecular dynamics trajectory to an external disk file

PRTERR Subroutine

"prterr" writes out a set of coordinates to a disk file prior to aborting on a serious error

PRTINT Subroutine

"prtint" writes out a set of Z-matrix internal coordinates to an external disk file

PRTMOL2 Program

"prtmol2" writes out a set of coordinates in Sybyl MOL2 format to an external disk file

PRTPDB Subroutine

"prtpdb" writes out a set of Protein Data Bank coordinates to an external disk file

PRTPRM Subroutine

"prtprm" writes out a formatted listing of the default set of potential energy parameters for a force field

PRTSEQ Subroutine

"prtseq" writes out a biopolymer sequence to an external disk file with 50 residues per line and distinct chains separated by blank lines

PRTXMOL Subroutine

"prtxmol" writes out a set of Cartesian coordinates for all active atoms in the XMOL program's generic XYZ format

PRTXYZ Subroutine

"prtxyz" writes out a set of Cartesian coordinates to an external disk file

PSS Program

"pss" implements the potential smoothing plus search method for global optimization in Cartesian coordinate space with local searches performed in Cartesian or torsional space

PSS1 Function

"pss1" is a service routine that computes the energy and gradient during PSS global optimization in Cartesian coordinate space

PSS2 Subroutine

"pss2" is a service routine that computes the sparse matrix Hessian elements during PSS global optimization in Cartesian coordinate space

PSSMIN Subroutine

"pssmin" is used during the potential smoothing and search procedure to perform a local optimization at the current smoothing level

PSSRGD1 Function

"pssrgd1" is a service routine that computes the energy and gradient during PSS global optimization over rigid bodies

PSSRIGID Program

"pssrigid" implements the potential smoothing plus search method for global optimization for a set of rigid bodies

PSSROT Program

"pssrot" implements the potential smoothing plus search method for global optimization in torsional space

PSSROT1 Function

"pssrot1" is a service routine that computes the energy and gradient during PSS global optimization in torsional space

PSSWRITE Subroutine

PTINCY Function

PZEXTR Subroutine

QRFACT Subroutine

"qrfact" performs Householder transformations with column pivoting (optional) to compute a QR factorization of the m by n matrix a; the routine determines an orthogonal matrix q, a permutation matrix p, and an upper trapezoidal matrix r with diagonal elements of nonincreasing magnitude, such that $a*p = q*r$; the Householder transformation for column k, $k = 1,2,...,min(m,n)$, is of the form

QRSOLVE Subroutine

"qrsolve" solves $a*x=b$ and $d*x=0$ in the least squares sense; normally used in combination with routine "qrfact" to solve least squares problems

QUATFIT Subroutine

"quatfit" uses a quaternion-based method to achieve the best fit superposition of two sets of coordinates

RANDOM Function

"random" generates a random number on [0,1] via a long period generator due to L'Ecuyer with Bays-Durham shuffle

RANVEC Subroutine

"ranvec" generates a unit vector in 3-dimensional space with uniformly distributed random orientation

RATTLE Subroutine

"rattle" implements the first portion of the rattle algorithm by correcting atomic positions and half-step velocities to maintain constrained interatomic distances

RATTLE2 Subroutine

"rattle2" implements the second portion of the rattle algorithm by correcting the full-step velocities in order to maintain constrained interatomic distances

READBLK Subroutine

"readblk" reads in a set of coordinate files and transfers the coordinates to internal arrays for use in the computation of time correlation functions

READDYN Subroutine

"readdyn" get the positions, velocities and accelerations for a molecular dynamics restart from an external disk file

READINT Subroutine

"readint" gets a set of Z-matrix internal coordinates from an external file

READMOL2 Subroutine

"readmol2" gets a set of Sybyl MOL2 coordinates from an external disk file

READPDB Subroutine

"readpdb" gets a set of Protein Data Bank coordinates from an external disk file

READPRM Subroutine

"readprm" processes the potential energy parameter file in order to define the default force field parameters

READSEQ Subroutine

"readseq" gets a biopolymer sequence containing one or more separate chains from an external file; all lines containing sequence must begin with the starting sequence number, the actual sequence is read from subsequent nonblank characters

READXYZ Subroutine

"readxyz" gets a set of Cartesian coordinates from an external disk file

REFINE Subroutine

"refine" performs minimization of the atomic coordinates of an initial crude embedded distance geometry structure versus the bound, chirality, planarity and torsional error functions

RESTRAIN Subroutine

"restrain" defines any geometric restraint interaction terms to be included in the potential energy calculation

RFINDEX Subroutine

RGDSRCH Subroutine

RIBOSOME Subroutine

"ribosome" translates a polypeptide structure in Protein Data Bank format to a Cartesian coordinate file and sequence file

RIGIDXYZ Subroutine

"rigidxyz" computes Cartesian coordinates for a rigid body group via rotation and translation of reference coordinates

RINGS Subroutine

"rings" searches the structure for small rings and stores their component atoms; code to remove the reducible rings consisting of smaller rings is commented in this version since reducible rings are needed for parameter assignment

RMSERROR Subroutine

"rmserror" computes the maximum absolute deviation and the rms deviation from the distance bounds, and the number and rms value of the distance restraint violations

RMSFIT Function

"rmsfit" computes the rms fit of two coordinate sets

ROTANG Function

ROTCHECK Function

"rotcheck" tests a specified candidate rotatable bond for the disallowed case where inactive atoms are found on both sides of the candidate bond

ROTEULER Subroutine

"roteuler" computes a set of Euler angle values consistent with an input rotation matrix

ROTLIST Subroutine

"rotlist" generates the minimum list of all the atoms lying to one side of a pair of directly bonded atoms; optionally finds the minimal list by choosing the side with fewer atoms

ROTMAT Subroutine

"rotmat" find the rotation matrix that converts from the local coordinate system at each multipole site to the global system

ROTPOLE Subroutine

"rotpole" computes the atomic multipole values in the global coordinate frame by applying a rotation matrix to a set of locally defined multipoles

SADDLE Program

"saddle" finds a transition state between two conformational minima using a combination of ideas from the synchronous transit (Halgren-Lipscomb) and quadratic path (Bell-Crighton) methods

SADDLE1 Function

SADDLES Subroutine

"saddles" constructs circles, convex edges and saddle faces

SCAN Program

"scan" attempts to find all the local minima on a potential energy surface via an iterative series of local searches

SCAN1 Function

"scan1" is a service routine that computes the energy and gradient during exploration of a potential energy surface via iterative local search

SCAN2 Subroutine

"scan2" is a service routine that computes the sparse matrix Hessian elements during exploration of a potential energy surface via iterative local search

SDAREA Subroutine

"sdarea" scales the atomic friction coefficient of each atom based on its accessible area during stochastic dynamics

SDSTEP Subroutine

"sdstep" performs a single stochastic dynamics time step via a velocity Verlet integration algorithm

SDTERM Subroutine

"sdterm" gets frictional and random force terms needed to update positions and velocities via stochastic dynamics

SEARCH Subroutine

"search" is a line search minimizer based upon parabolic extrapolation and cubic interpolation using both function and gradient values; if forced to search in an uphill direction, return is after the initial step

SETIME Subroutine

"setime" initializes the elapsed interval CPU timer

SHAKEUP Subroutine

"shakeup" initializes any holonomic constraints for use with the rattle algorithm during molecular dynamics

SIDECHAIN Subroutine

"sidechain" builds the side chain for a single amino acid residue in terms of internal coordinates

SIGMOID Function

"sigmoid" implements a normalized sigmoidal function on the interval [0,1]; the curves connect (0,0) to (1,1) and have a cooperativity controlled by beta, they approach a straight line as beta -> 0 and get more nonlinear as beta increases

SLATER Subroutine

"slater" is a general routine for computing the overlap integrals between two Slater-type orbitals

SMOOTH Subroutine

"smooth" sets extent of potential surface deformation for use with potential smoothing plus search, the diffusion equation method or Gaussian density annealing

SNIFFER Program

"sniffer" performs a global energy minimization using a discrete version of Griewank's global search trajectory

SNIFFER1 Function

"sniffer1" is a service routine that computes the energy and gradient for the Sniffer global optimization method

SOAK Subroutine

"soak" takes a currently defined solute system and places it into a solvent box, with removal of any solvent molecules that overlap the solute

SOLVATE Subroutine

"solvate" assigns macroscopic solvation energy parameters for the Eisenberg-McLachlan ASP, Ooi-Scheraga SASA or Macromodel GB/SA solvation models

SORT Subroutine

"sort" takes an input list of integers and sorts it into ascending order using the Heapsort algorithm

SORT2 Subroutine

"sort2" takes an input list of reals and sorts it into ascending order using the Heapsort algorithm; it also returns a key into the original ordering

SORT3 Subroutine

"sort3" takes an input list of integers and sorts it into ascending order using the Heapsort algorithm; it also returns a key into the original ordering

SORT4 Subroutine

"sort4" takes an input list of integers and sorts it into ascending absolute value using the Heapsort algorithm

SORT5 Subroutine

"sort5" takes an input list of integers and sorts it into ascending order based on each value modulo "m"

SORT6 Subroutine

"sort6" takes an input list of character strings and sorts it into alphabetical order using the Heapsort algorithm

SORT7 Subroutine

"sort7" takes an input list of character strings and sorts it into alphabetical order using the Heapsort algorithm; it also returns a key into the original ordering

SPACEFILL Program

"spacefill" computes the surface area and volume of a structure; the van der Waals, accessible-excluded, and contact-reentrant definitions are available

SQUARE Subroutine

"square" is a nonlinear least squares routine derived from the IMSL routine BCLSF and More's Minpack routine LMDER; the Jacobian is estimated by finite differences and bounds can be specified for the variables to be refined

SUFFIX Subroutine

"suffix" checks a filename for the presence of an extension, and appends an extension if none is found

SUPERPOSE Program

"superpose" takes two input structures and superimposes them in the optimal least squares sense; it will attempt to match up all pairs of atoms or just those specified by the user

SURFACE Subroutine

"surface" performs an analytical computation of the weighted solvent accessible surface area of each atom and the first derivatives of the area with respect to Cartesian coordinates

SURFATOM Subroutine

"surfatom" performs an analytical computation of the surface area of a specified atom; a simplified version of "surface"

SWITCH Subroutine

"switch" sets the coeffcients used by the fifth and seventh order polynomial switching functions for spherical cutoffs

SYBYLXYZ Program

"sybylxyz" takes as input a Sybyl MOL2 coordinates file, converts to and then writes out Cartesian coordinates

SYMMETRY Subroutine

"symmetry" applies symmetry operators to the fractional coordinates of the asymmetric unit in order to generate the symmetry related atoms of the full unit cell

TANGENT Subroutine

"tangent" finds the projected gradient on the synchronous transit path for a point along the transit pathway

TEMPER Subroutine

"temper" maintains a constant desired temperature by scaling the velocities via coupling to an external temperature bath

TESTGRAD Program

"testgrad" computes and compares the analytical and numerical gradient vectors of the potential energy function with respect to Cartesian coordinates

TESTHESS Program

"testhess" computes and compares the analytical and numerical Hessian matrices of the potential energy function with respect to Cartesian coordinates

TESTLIGHT Program

"testlight" performs a set of timing tests to compare the evaluation of potential energy and energy/gradient using the method of lights with a double loop over all atom pairs

TESTROT Program

"testrot" computes and compares the analytical and numerical gradient vectors of the potential energy function with respect to rotatable torsional angles

TIMER Program

"timer" measures the CPU time required for file reading and parameter assignment, potential energy computation, energy and gradient computation, and Hessian matrix evaluation

TIMEROT Program

"timerot" measures the CPU time required for file reading and parameter assignment, potential energy computation, energy and gradient over torsions, and torsional angle Hessian matrix evaluation

TNCG Subroutine

"tncg" implements a truncated Newton optimization algorithm in which a preconditioned linear conjugate gradient method is used to approximately solve Newton's equations; special features include use of an explicit sparse Hessian or finite-difference gradient-Hessian products within the PCG iteration; the exact Newton search directions can be used optionally; by default the algorithm checks for negative curvature to prevent convergence to a stationary point having negative eigenvalues; if a saddle point is desired this test can be removed by disabling "negtest"

TNSOLVE Subroutine

"tnsolve" uses a linear conjugate gradient method to find an approximate solution to the set of linear equations represented in matrix form by Hp = -g (Newton's equations)

TORPHASE Subroutine

"torphase" sets the n-fold amplitude and phase values for each torsion via sorting of the input parameters

TORSER Function

"torser" computes the torsional error function and its first derivatives with respect to the atomic Cartesian coordinates based on the deviation of specified torsional angles from desired values, the contained bond angles are also restrained to avoid a numerical instability

TORSIONS Subroutine

"torsions" finds the total number of dihedral angles and the numbers of the four atoms defining each dihedral angle

TORUS Subroutine

"torus" sets a list of all of the temporary torus positions by testing for a torus between each atom and its neighbors

TOTERR Function

"toterr" is the error function and derivatives for a distance geometry embedding; it includes components from the distance bounds, hard sphere contacts, local geometry, chirality and torsional restraint errors

TRANSIT Function

"transit" evaluates the synchronous transit function and gradient; linear and quadratic transit paths are available

TRIANGLE Subroutine

"triangle" smooths the upper and lower distance bounds via the triangle inequality using a full-matrix variant of the Floyd-Warshall shortest path algorithm; this routine is usually much slower than the sparse matrix shortest path methods in "geodesic" and "trifix", and should be used only for comparison with answers generated by those routines

TRIFIX Subroutine

"trifix" rebuilds both the upper and lower distance bound matrices following tightening of one or both of the bounds between a specified pair of atoms, "p" and "q", using a modification of Murchland's shortest path update algorithm

TRIMTEXT Function

"trimtext" finds and returns the location of the last non-blank character before the first null character in an input text string; the function returns zero if no such character is found

TRIPLE Function

"triple" finds the triple product of three vectors; used as a service routine by the Connolly surface area and volume computation

TRUST Subroutine

"trust" updates the model trust region for a nonlinear least squares calculation; this version is based on the ideas found in NL2SOL and in Dennis and Schnabel's book

UDIRECT Subroutine

"udirect" evaluates the electric field at a polarizable atom due to permanent atomic multipoles at a second atom, and vice versa, for use in computation of direct induced dipole moments

UMUTUAL Subroutine

"umutual" evaluates the electric field at a polarizable atom due to the induced atomic dipoles at a second atom, and vice versa, for use in computation of mutual induced dipole moments

UNITCELL Subroutine

"unitcell" gets the periodic boundary box size and related values from an external keyword file

UPCASE Subroutine

"upcase" converts a text string to all upper case letters

VAM Subroutine

"vam" takes the analytical molecular surface defined as a collection of spherical and toroidal polygons and uses it to compute the volume and surface area

VCROSS Subroutine

"vcross" finds the cross product of two vectors

VDWERR Function

"vdwerr" is the hard sphere van der Waals bound error function and derivatives that penalizes close nonbonded contacts, pairwise neighbors are generated via the method of lights

VECANG Function

"vecang" finds the angle between two vectors handed with respect to a coordinate axis; returns an angle in the range [0,2*pi]

VERLET Subroutine

"verlet" performs a single molecular dynamics time step by means of the velocity Verlet multistep recursion formula

VERSION Subroutine

"version" checks the name of a file about to be opened; if if "old" status is passed, the name of the highest current version is returned; if "new" status is passed the filename of the next available unused version is generated

VIBRATE Program

"vibrate" performs a vibrational normal mode analysis; the Hessian matrix of second derivatives is determined and then diagonalized both directly and after mass weighting; output consists of the eigenvalues of the force constant matrix as well as the vibrational frequencies and displacements

VIBROT Program

VNORM Subroutine

"vnorm" normalizes a vector to unit length; used as a service routine by the Connolly surface area and volume computation

VOLUME Subroutine

"volume" calculates the excluded volume via the Connolly analytical volume and surface area algorithm

VOLUME1 Subroutine

"volume1" calculates first derivatives of the total excluded volume with respect to the Cartesian coordinates of each atom

VOLUME2 Subroutine

"volume2" calculates second derivatives of the total excluded volume with respect to the Cartesian coordinates of the atoms

WRITEOUT Subroutine

"writeout" is used by each of the optimization routines to save imtermediate atomic coordinates to a disk file

XTALERR Subroutine

XTALFIT Program

"xtalfit" computes an optimized set of potential energy parameters for user specified van der Waals and electrostatic interactions by fitting to crystal structure, lattice energy and monomer dipole moment data

XTALLAT1 Function

XTALMIN Program

"xtalmin" performs a full crystal energy minimization by alternating cycles of truncated Newton optimization over atomic coordinates with variable metric optimization over the six lattice dimensions and angles

XTALMOL1 Function

XTALMOL2 Subroutine

XTALMOVE Subroutine

XTALPRM Subroutine

"xtalprm" stores or retrieves a crystal structure; used to make a previously stored structure the currently active structure, or to store a structure for later use; only the intermolecular energy terms are provided for

XTALWRT Subroutine

XYZATM Subroutine

"xyzatm" computes the Cartesian coordinates of a single atom from its defining internal coordinate values

XYZEDIT Program

"xyzedit" provides for modification and manipulation of the contents of a Cartesian coordinates file

XYZINT Program

"xyzint" takes as input a Cartesian coordinates file, then converts to and writes out an internal coordinates file

XYZPDB Program

"xyzpdb" takes as input a Cartesian coordinates file, then converts to and writes out a Protein Data Bank file

XYZRIGID Subroutine

"xyzrigid" computes the center of mass and Euler angle rigid body coordinates for each atom group in the system

XYZSYBYL Program

"xyzsybyl" takes as input a Cartesian coordinates file, converts to and then writes out a Sybyl MOL2 file

ZATOM Subroutine

"zatom" adds an atom to the end of the current Z-matrix and then increments the atom counter; atom type, defining atoms and internal coordinates are passed as arguments

ZHELP Subroutine

"zhelp" prints the general information and instructions for the Z-matrix editing program "intedit"

ZVALUE Subroutine

"zvalue" gets user supplied values for selected coordinates as needed by the internal coordinate editing program

10. Contents of Common Block Variables

The Fortran common blocks found in the TINKER package are listed below along with a brief description of the contents of each variable in each common block. Each individual common block is present as a separate ".i" file in the /source subdirectory. A source code listing containing each of the source code modules and each of the common blocks can be produced by running the "listing.make" script found in the distribution.

ACTION   **total number of each energy term computed**

neb   number of bond stretch energy terms computed
nea   number of angle bend energy terms computed
neba   number of stretch-bend energy terms computed
neub   number of Urey-Bradley energy terms computed
neaa   number of angle-angle energy terms computed
neopb   number of out-of-plane bend energy terms computed
neid   number of improper dihedral energy terms computed
neit   number of improper torsion energy terms computed
net   number of torsional energy terms computed
nebt   number of stretch-torsion energy terms computed
nett   number of torsion-torsion energy terms computed
nev   number of van der Waals energy terms computed
ne14   number of 1-4 van der Waals energy terms computed
nec   number of charge-charge energy terms computed
necd   number of charge-dipole energy terms computed
ned   number of dipole-dipole energy terms computed
nem   number of multipole energy terms computed
nep   number of polarization energy terms computed
new   number of Ewald summation energy terms computed
ner   number of reaction field energy terms computed
nes   number of solvation energy terms computed
neg   number of geometric restraint energy terms computed
nex   number of extra energy terms computed

ALIGN   **information for superposition of structures**

wfit   weights assigned to atom pairs during superposition
nfit   number of atoms to use in superimposing two structures
ifit   atom numbers of pairs of atoms to be superimposed

AMINOS   **standard abbreviations for amino acid types**

amino   three-letter abbreviations for amino acids and caps
amino1   one-letter abbreviations for amino acids and caps

ANALYZ   **energy components partitioned over atoms**

aeb   bond stretch energy partitioned over atoms
aea   angle bend energy partitioned over atoms
aeba   stretch-bend energy partitioned over atoms
aeub   Urey-Bradley energy partitioned over atoms

aeaa	angle-angle energy partitioned over atoms
aeopb	out-of-plane bend energy partitioned over atoms
aeid	improper dihedral energy partitioned over atoms
aeit	improper torsion energy partitioned over atoms
aet	torsional energy partitioned over atoms
aebt	stretch-torsion energy partitioned over atoms
aett	torsion-torsion energy partitioned over atoms
aev	van der Waals energy partitioned over atoms
ae14	1-4 van der Waals energy partitioned over atoms
aec	charge-charge energy partitioned over atoms
aecd	charge-dipole energy partitioned over atoms
aed	dipole-dipole energy partitioned over atoms
aem	multipole energy partitioned over atoms
aep	polarization energy partitioned over atoms
aew	Ewald summation energy partitioned over atoms
aer	reaction field energy partitioned over atoms
aes	solvation energy partitioned over atoms
aeg	geometric restraint energy partitioned over atoms
aex	extra energy term partitioned over atoms

ANGANG   **angle-angle terms in current structure**

kaa	force constant for angle-angle cross terms
nangang	total number of angle-angle interactions
iaa	angle numbers used in each angle-angle term

ANGLE   **bond angles within the current structure**

ak	harmonic angle force constant (kcal/mole/rad**2)
anat	ideal bond angle or phase shift angle (degrees)
afld	periodicity for Fourier bond angle term
nangle	total number of bond angles in the system
iang	numbers of the atoms in each bond angle
angtyp	potential energy function type for each bond angle

ANGPOT   **specifics of bond angle functional form**

angunit	convert angle force constant to kcal/mole/deg**2
cang	cubic term in angle bending potential
qang	quartic term in angle bending potential
pang	quintic term in angle bending potential
sang	sextic term in angle bending potential
aaunit	convert angle-angle constant to kcal/mole/deg**2
opbunit	convert out-plane bend constant to kcal/mole/deg**2
stbnunit	convert str-bnd constant to kcal/mole/deg-Ang**2

ARGUE   **command line arguments at program startup**

maxarg	maximum number of command line arguments
narg	number of command line arguments to the program
arg	strings containing the command line arguments
listarg	flag to mark available command line arguments

ATMLST   **local geometry terms involving each atom**

bndlist   list of the bond numbers involving each atom
anglist   list of the angle numbers centered on each atom

ATMTYP   **atomic properties for each current atom**

mass   atomic weight for each atom in the system
tag   integer atom labels from input coordinates file
class   atom class number for each atom in the system
atomic   atomic number for each atom in the system
valence   valence number for each atom in the system
name   atom name for each atom in the system
story   descriptive type for each atom in system

ATOMS   **number, position and type of current atoms**

x   current x-coordinate for each atom in the system
y   current y-coordinate for each atom in the system
z   current z-coordinate for each atom in the system
n   total number of atoms in the current system
type   atom type number for each atom in the system

BATH   **temperature and pressure bath control values**

kelvin   target value for the system temperature (K)
atmsph   target value for the system pressure (atm)
tautemp   time constant in psec for temperature bath coupling
taupres   time constant in psec for pressure bath coupling
compress   isothermal compressibility of medium (atm-1)
isothermal   logical flag geverning use of temperature bath
isobaric   logical flag governing use of pressure bath

BNDPOT   **specifics of bond stretch functional form**

bndunit   convert bond force constant to kcal/mole/Ang**2
cbnd   cubic term in bond stretch potential
qbnd   quartic term in bond stretch potential
bndtyp   type of bond stretch potential energy function

BOND   **covalent bonds in the current structure**

bk   bond stretch force constants (kcal/mole/Ang**2)
bl   ideal bond length values in Angstroms
nbond   total number of bond stretches in the system
ibnd   numbers of the atoms in each bond stretch

BOUND   **control of periodic boundary conditions**

use_bounds   flag to use periodic boundary conditions
use_image   flag to use images for periodic system

use_replica	flag to use replicates for periodic system

BOXES   **parameters for periodic boundary conditions**

xbox	length in Angs of a-axis of periodic box
ybox	length in Angs of b-axis of periodic box
zbox	length in Angs of c-axis of periodic box
alpha	angle in degrees between b- and c-axes of box
beta	angle in degrees between a- and c-axes of box
gamma	angle in degrees between a- and b-axes of box
xbox2	half of the a-axis length of periodic box
ybox2	half of the b-axis length of periodic box
zbox2	half of the c-axis length of periodic box
box34	three-fourths axis length of truncated octahedron
recip	reciprocal lattice vectors as matrix columns
volbox	volume in Angs**3 of the periodic box
beta_sin	sine of the beta periodic box angle
beta_cos	cosine of the beta periodic box angle
gamma_sin	sine of the gamma periodic box angle
gamma_cos	cosine of the gamma periodic box angle
beta_term	term used in generating triclinic box
gamma_term	term used in generating triclinic box
orthogonal	flag to mark periodic box as orthogonal
monoclinic	flag to mark periodic box as monoclinic
triclinic	flag to mark periodic box as triclinic
octahedron	flag to mark box as truncated octahedron
spacegrp	space group symbol for the unitcell type

CELL   **periodic boundaries using replicated cells**

xcell	length of the a-axis of the complete replicated cell
ycell	length of the b-axis of the complete replicated cell
zcell	length of the c-axis of the complete replicated cell
xcell2	half the length of the a-axis of the replicated cell
ycell2	half the length of the b-axis of the replicated cell
zcell2	half the length of the c-axis of the replicated cell
ncell	total number of cell replicates for periodic boundaries
icell	offset along axes for each replicate periodic cell

CENTRE   **coordinates relative to molecule centroid**

xcm	x-offset of atom from molecular center of mass
ycm	y-offset of atom from molecular center of mass
zcm	z-offset of atom from molecular center of mass

CHARGE   **partial charges for the current structure**

pchg	magnitude of the partial charges (e-)
nion	total number of partial charges in system
iion	number of the atom site for each partial charge
jion	neighbor generation site for each partial charge
kion	cutoff switching site for each partial charge

CHGPOT   **specifics of electrostatics functional form**

dielec	dielectric constant for electrostatic interactions
chgscale	factor by which 1-4 electrostatic terms are scaled
chg12use	usage of 1-2 electrostatics (0=use, 1=omit, -1=scale)
chg13use	usage of 1-3 electrostatics (0=use, 1=omit, -1=scale)
chg14use	usage of 1-4 electrostatics (0=use, 1=omit, -1=scale)
neutnbr	logical flag governing use of neutral group neighbors
neutcut	logical flag governing use of neutral group cutoffs

CHRONO   **timing statistics for the current program**

cputim	elapsed cpu time in seconds since start of program

COUPLE   **near-neighbor atom connectivity lists**

n12	number of atoms directly bonded to each atom
i12	atom numbers of atoms 1-2 connected to each atom
n13	number of atoms in a 1-3 relation to each atom
i13	atom numbers of atoms 1-3 connected to each atom
n14	number of atoms in a 1-4 relation to each atom
i14	atom numbers of atoms 1-4 connected to each atom

CUTOFF   **cutoff distances for energy interactions**

vdwcut	cutoff distance for van der Waals interactions
chgcut	cutoff distance for charge-charge interactions
dplcut	cutoff distance for dipole-dipole interactions
vdwtaper	distance at which van der Waals switching begins
chgtaper	distance at which charge-charge switching begins
dpltaper	distance at which dipole-dipole switching begins
pmecut	cutoff distance for direct space Ewald summation
use_lights	flag to use method of lights neighbor generation

DERIV   **Cartesian coordinate derivative components**

deb	bond stretch Cartesian coordinate derivatives
dea	angle bend Cartesian coordinate derivatives
deba	stretch-bend Cartesian coordinate derivatives
deub	Urey-Bradley Cartesian coordinate derivatives
deaa	angle-angle Cartesian coordinate derivatives
deopb	out-of-plane bend Cartesian coordinate derivatives
deid	improper dihedral Cartesian coordinate derivatives
deit	improper torsion Cartesian coordinate derivatives
det	torsional Cartesian coordinate derivatives
debt	stretch-torsion Cartesian coordinate derivatives
dett	torsion-torsion Cartesian coordinate derivatives
dev	van der Waals Cartesian coordinate derivatives
de14	1-4 van der Waals Cartesian coordinate derivatives
dec	charge-charge Cartesian coordinate derivatives
decd	charge-dipole Cartesian coordinate derivatives

ded	dipole-dipole Cartesian coordinate derivatives
dem	multipole Cartesian coordinate derivatives
dep	polarization Cartesian coordinate derivatives
dew	Ewald summation Cartesian coordinate derivatives
der	reaction field Cartesian coordinate derivatives
des	solvation Cartesian coordinate derivatives
deg	geometric restraint Cartesian coordinate derivatives
dex	extra energy term Cartesian coordinate derivatives

DIPOLE   **atom & bond dipoles for current structure**

bdpl	magnitude of each of the dipoles (Debyes)
sdpl	position of each dipole between defining atoms
ndipole	total number of dipoles in the system
idpl	numbers of atoms that define each dipole

DISGEO   **distance geometry bounds and parameters**

bnd	distance geometry upper and lower bounds matrix
vdwrad	hard sphere radii for distance geometry atoms
vchir	signed volume values for chirality constraints
compact	index of local distance compaction on embedding
pathmax	maximum value of upper bound after smoothing
vdwmax	maximum value of hard sphere sum for an atom pair
nchir	total number of chirality constraints
ichir	numbers of atoms in each chirality constraint
use_invert	flag to use enantiomer closest to input structure
use_anneal	flag to use simulated annealing refinement

DOMEGA   **derivative components over dihedrals**

teb	bond stretch derivatives over torsions
tea	angle bend derivatives over torsions
teba	stretch-bend derivatives over torsions
teub	Urey-Bradley derivatives over torsions
teaa	angle-angle derivatives over torsions
teopb	out-of-plane bend derivatives over torsions
teid	improper dihedral derivatives over torsions
teit	improper torsion derivatives over torsions
tet	torsional derivatives over torsions
tebt	stretch-torsion derivatives over torsions
tett	torsion-torsion derivatives over torsions
tev	van der Waals derivatives over torsions
te14	1-4 van der Waals derivatives over torsions
tec	charge-charge derivatives over torsions
tecd	charge-dipole derivatives over torsions
ted	dipole-dipole derivatives over torsions
tem	atomic multipole derivatives over torsions
tep	polarization derivatives over torsions
tew	Ewald summation derivatives over torsions
ter	reaction field derivatives over torsions
tes	solvation derivatives over torsions

teg	geometric restraint derivatives over torsions
tex	extra energy term derivatives over torsions

ENERGI	**individual potential energy components**
eb	bond stretch potential energy of the system
ea	angle bend potential energy of the system
eba	stretch-bend potential energy of the system
eub	Urey-Bradley potential energy of the system
eaa	angle-angle potential energy of the system
eopb	out-of-plane bend potential energy of the system
eid	improper dihedral potential energy of the system
eit	improper torsion potential energy of the system
et	torsional potential energy of the system
ebt	stretch-torsion potential energy of the system
ett	torsion-torsion potential energy of the system
ev	van der Waals potential energy of the system
e14	1-4 van der Waals potential energy of the system
ec	charge-charge potential energy of the system
ecd	charge-dipole potential energy of the system
ed	dipole-dipole potential energy of the system
em	atomic multipole potential energy of the system
ep	polarization potential energy of the system
ew	Ewald summation potential energy of the system
er	reaction field potential energy of the system
es	solvation potential energy of the system
eg	geometric restraint potential energy of the system
ex	extra term potential energy of the system

FACES	**variables for Connolly's area and volume**
maxnbr	maximum number of neighboring atom pairs
maxtt	maximum number of temporary tori
maxt	maximum number of total tori
maxp	maximum number of probe positions
maxv	maximum number of vertices
maxen	maximum number of concave edges
maxfn	maximum number of concave faces
maxc	maximum number of circles
maxep	maximum number of convex edges
maxfs	maximum number of saddle faces
maxcy	maximum number of cycles
mxcyep	maximum number of cycle convex edges
maxfp	maximum number of convex faces
mxfpcy	maximum number of convex face cycles

FIELDS	**molecular mechanics force field description**
biotyp	force field atom type of each biopolymer type
forcefield	string used to describe the current forcefield

FILES	**name and number of current structure files**

nprior	number of previously existing cycle files
ldir	length in characters of the directory name
leng	length in characters of the base filename
filename	base filename used by default for all files
outfile	output filename used for intermediate results

FRACS   **atom distances to molecular center of mass**

xfrac	fractional coordinate along a-axis of center of mass
yfrac	fractional coordinate along b-axis of center of mass
zfrac	fractional coordinate along c-axis of center of mass

GROUP   **partitioning of system into atom groups**

wgrp	energetic weight of a group-group interaction
grpnum	original group number for each nonempty group
ngrp	total number of atom groups in the system
kgrp	contiguous list of the atoms in each group
igrp	first and last atom of each group in the list
grplist	number of the group to which each atom belongs
use_group	flag to use partitioning of system into groups

HESCUT   **cutoff value for Hessian matrix elements**

| hesscut | magnitude of smallest allowed Hessian element |

HESSN   **Cartesian Hessian elements for a single atom**

hessx	Hessian elements for x-component of current atom
hessy	Hessian elements for y-component of current atom
hessz	Hessian elements for z-component of current atom

IMPROP   **improper dihedrals in the current structure**

kprop	force constant values for improper dihedral angles
vprop	ideal improper dihedral angle value in degrees
niprop	total number of improper dihedral angles in the system
iiprop	numbers of the atoms in each improper dihedral angle

IMPTOR   **improper torsions in the current structure**

itors1	1-fold amplitude and phase for each improper torsion
itors2	2-fold amplitude and phase for each improper torsion
itors3	3-fold amplitude and phase for each improper torsion
nitors	total number of improper torsional angles in the system
iitors	numbers of the atoms in each improper torsional angle

INFORM   **control flags for I/O and program flow**

| verbose | logical flag to turn on extra information |
| debug | logical flag to turn on full debug printing |

exitpause	logical flag to wait for carriage return on exit
abort	logical flag to stop execution at next chance

INTER   **sum of intermolecular energy components**

einter	total intermolecular potential energy

IOUNIT   **Fortran input/output (I/O) unit numbers**

iout	Fortran I/O unit for major output (default=6)
input	Fortran I/O unit for major input (default=5)

KANANG   **forcefield parameters for angle-angle terms**

anan	angle-angle cross term parameters for each atom class

KANGS   **forcefield parameters for bond angle bending**

maxna	maximum number of harmonic angle bend parameter entries
maxna5	maximum number of 5-membered ring angle bend entries
maxna4	maximum number of 4-membered ring angle bend entries
maxna3	maximum number of 3-membered ring angle bend entries
maxnaf	maximum number of Fourier angle bend parameter entries
acon	force constant parameters for harmonic angle bends
acon5	force constant parameters for 5-ring angle bends
acon4	force constant parameters for 4-ring angle bends
acon3	force constant parameters for 3-ring angle bends
aconf	force constant parameters for Fourier angle bends
ang	bond angle parameters for harmonic angle bends
ang5	bond angle parameters for 5-ring angle bends
ang4	bond angle parameters for 4-ring angle bends
ang3	bond angle parameters for 3-ring angle bends
angf	phase shift angle and periodicity for Fourier bends
ka	string of atom classes for harmonic angle bends
ka5	string of atom classes for 5-ring angle bends
ka4	string of atom classes for 4-ring angle bends
ka3	string of atom classes for 3-ring angle bends
kaf	string of atom classes for Fourier angle bends

KATOMS   **forcefield parameters for the atom types**

weight	average atomic mass of each atom type
atmcls	atom class number for each of the atom types
atmnum	atomic number for each of the atom types
ligand	number of atoms to be attached to each atom type
symbol	modified atomic symbol for each atom type
describe	string identifing each of the atom types

KBONDS   **forcefield parameters for bond stretching**

maxnb	maximum number of harmonic bond stretch parameter entries

maxnb5	maximum number of 5-membered ring bond stretch entries
maxnb4	maximum number of 4-membered ring bond stretch entries
maxnb3	maximum number of 3-membered ring bond stretch entries
bcon	force constant parameters for harmonic bond stretch
bcon5	force constant parameters for 5-ring bond stretch
bcon4	force constant parameters for 4-ring bond stretch
bcon3	force constant parameters for 3-ring bond stretch
blen	bond length parameters for harmonic bond stretch
blen5	bond length parameters for 5-ring bond stretch
blen4	bond length parameters for 4-ring bond stretch
blen3	bond length parameters for 3-ring bond stretch
kb	string of atom classes for harmonic bond stretch
kb5	string of atom classes for 5-ring bond stretch
kb4	string of atom classes for 4-ring bond stretch
kb3	string of atom classes for 3-ring bond stretch

KCHRGE   **forcefield parameters for partial charges**

chg	partial charge parameters for each atom type

KDIPOL   **forcefield parameters for bond dipoles**

maxnd	maximum number of bond dipole parameter entries
maxnd5	maximum number of 5-membered ring dipole entries
maxnd4	maximum number of 4-membered ring dipole entries
maxnd3	maximum number of 3-membered ring dipole entries
dpl	dipole moment parameters for bond dipoles
dpl5	dipole moment parameters for 5-ring dipoles
dpl4	dipole moment parameters for 4-ring dipoles
dpl3	dipole moment parameters for 3-ring dipoles
pos	dipole position parameters for bond dipoles
pos5	dipole position parameters for 5-ring dipoles
pos4	dipole position parameters for 4-ring dipoles
pos3	dipole position parameters for 3-ring dipoles
kd	string of atom classes for bond dipoles
kd5	string of atom classes for 5-ring dipoles
kd4	string of atom classes for 4-ring dipoles
kd3	string of atom classes for 3-ring dipoles

KEYS   **contents of current keyword parameter file**

nkey	number of nonblank lines in the keyword file
keyline	contents of each individual keyword file line

KHBOND   **forcefield parameters for H-bonding terms**

maxnhb	maximum number of hydrogen bonding pair entries
radhb	radius parameter for hydrogen bonding pairs
epshb	well depth parameter for hydrogen bonding pairs
khb	string of atom types for hydrogen bonding pairs

KIPROP   **forcefield parameters for improper dihedral**

maxndi	maximum number of improper dihedral parameter entries
dcon	force constant parameters for improper dihedrals
tdi	ideal dihedral angle values for improper dihedrals
kdi	string of atom classes for improper dihedral angles

KITORS   **forcefield parameters for improper torsions**

maxnti	maximum number of improper torsion parameter entries
ti1	torsional parameters for improper 1-fold rotation
ti2	torsional parameters for improper 2-fold rotation
ti3	torsional parameters for improper 3-fold rotation
kti	string of atom classes for improper torsional parameters

KMULTI   **forcefield parameters for atomic multipoles**

maxnmp	maximum number of atomic multipole parameter entries
multip	atomic monopole, dipole and quadrupole values
mpaxis	type of local axis definition for atomic multipoles
kmp	string of atom types for atomic multipoles

KOPBND   **forcefield parameters for out-of-plane bend**

maxnopb	maximum number of out-of-plane bending entries
copb	force constant parameters for out-of-plane bending
kaopb	string of atom classes for out-of-plane bending

KORBS   **forcefield parameters for pi-system orbitals**

maxnpi	maximum number of pi-system bond parameter entries
electron	number of pi-electrons for each atom class
ionize	ionization potential for each atom class
repulse	repulsion integral value for each atom class
sslope	slope for bond stretch vs. pi-bond order
tslope	slope for 2-fold torsion vs. pi-bond order
kpi	string of atom classes for pi-system bonds

KPOLR   **forcefield parameters for polarizability**

polr	dipole polarizability parameters for each atom type

KSTBND   **forcefield parameters for stretch-bending**

stbn	stretch-bending parameters for each atom class

KSTTOR   **forcefield parameters for stretch-torsions**

maxnbt	maximum number of stretch-torsion parameter entries
btcon	force constant parameters for stretch-torsion
kbt	string of atom classes for bonds in stretch-torsion

KTORSN   **forcefield parameters for torsional angles**

maxnt	maximum number of torsional angle parameter entries
maxnt5	maximum number of 5-membered ring torsion entries
maxnt4	maximum number of 4-membered ring torsion entries
t1	torsional parameters for standard 1-fold rotation
t2	torsional parameters for standard 2-fold rotation
t3	torsional parameters for standard 3-fold rotation
t4	torsional parameters for standard 4-fold rotation
t5	torsional parameters for standard 5-fold rotation
t6	torsional parameters for standard 6-fold rotation
t15	torsional parameters for 1-fold rotation in 5-ring
t25	torsional parameters for 2-fold rotation in 5-ring
t35	torsional parameters for 3-fold rotation in 5-ring
t45	torsional parameters for 4-fold rotation in 5-ring
t55	torsional parameters for 5-fold rotation in 5-ring
t65	torsional parameters for 6-fold rotation in 5-ring
t14	torsional parameters for 1-fold rotation in 4-ring
t24	torsional parameters for 2-fold rotation in 4-ring
t34	torsional parameters for 3-fold rotation in 4-ring
t44	torsional parameters for 4-fold rotation in 4-ring
t54	torsional parameters for 5-fold rotation in 4-ring
t64	torsional parameters for 6-fold rotation in 4-ring
kt	string of atom classes for torsional angles
kt5	string of atom classes for 5-ring torsions
kt4	string of atom classes for 4-ring torsions

KURYBR   **forcefield parameters for Urey-Bradley terms**

maxnu	maximum number of Urey-Bradley parameter entries
ucon	force constant parameters for Urey-Bradley terms
dst13	ideal 1-3 distance parameters for Urey-Bradley terms
ku	string of atom classes for Urey-Bradley terms

KVDWPR   **forcefield parameters for special vdw terms**

maxnvp	maximum number of special van der Waals pair entries
radpr	radius parameter for special van der Waals pairs
epspr	well depth parameter for special van der Waals pairs
kvpr	string of atom classes for special van der Waals pairs

KVDWS   **forcefield parameters for van der Waals terms**

rad	van der Waals radius parameter for each atom class
eps	van der Waals well depth parameter for each atom class
rad4	van der Waals radius parameter in 1-4 interactions
eps4	van der Waals well depth parameter in 1-4 interactions
reduct	van der Waals reduction factor for each atom class

LIGHT   **indices for method of lights pair neighbors**

nlight	total number of sites for method of lights calculation
kbx	low index of neighbors of each site in the x-sorted list

kby	low index of neighbors of each site in the y-sorted list
kbz	low index of neighbors of each site in the z-sorted list
kex	high index of neighbors of each site in the x-sorted list
key	high index of neighbors of each site in the y-sorted list
kez	high index of neighbors of each site in the z-sorted list
locx	pointer from x-sorted list into original interaction list
locy	pointer from y-sorted list into original interaction list
locz	pointer from z-sorted list into original interaction list
rgx	pointer from original interaction list into x-sorted list
rgy	pointer from original interaction list into y-sorted list
rgz	pointer from original interaction list into z-sorted list

LINMIN   **parameters for line search minimization**

cappa	stringency of line search (0=tight < cappa < 1=loose)
stpmin	minimum step length in current line search direction
stpmax	maximum step length in current line search direction
angmax	maximum angle between search direction and -gradient
intmax	maximum number of cubic interpolation attempts

MATH   **mathematical and geometrical constants**

radian	conversion factor from radians to degrees
pi	numerical value of the geometric constant
sqrtpi	numerical value of the square root of Pi
twosix	numerical value of the sixth root of two
logten	numerical value of the natural log of ten

MINIMA   **general parameters for minimizations**

fctmin	value below which function is deemed optimized
hguess	initial value for the H-matrix diagonal elements
maxiter	maximum number of iterations during optimization
nextiter	iteration number to use for the first iteration
iprint	iterations between status printing (0=no printing)
iwrite	iterations between coordinate dumps (0=no dumps)

MOLCUL   **individual molecules within current system**

molmass	molecular weight for each molecule in the system
totmass	total weight of all the molecules in the system
nmol	total number of separate molecules in the system
kmol	contiguous list of the atoms in each molecule
imol	first and last atom of each molecule in the list
molcule	number of the molecule to which each atom belongs

MOLDYN   **velocity and acceleration on MD trajectory**

v	current velocity of each atom along the x,y,z-axes
a	current acceleration of each atom along x,y,z-axes
a_old	previous acceleration of each atom along x,y,z-axes

MOMENT   **components of the net multipole moments**

netchg   net electric charge on the total system
xdipole   dipole moment of the system along the x-axis
ydipole   dipole moment of the system along the y-axis
zdipole   dipole moment of the system along the z-axis

MPOLE   **multipole components for current structure**

maxpole   max components (monopole=1,dipole=4,quadrupole=13)
pole   multipole values for each site in the local frame
rpole   multipoles rotated to the global coordinate system
dpole   derivative rotation matrix for each multipole
npole   total number of multipole sites in the system
ipole   number of the atom for each multipole site
polsiz   number of mutipole components at each multipole site
zaxis   number of the z-axis defining atom for each site
xaxis   number of the x-axis defining atom for each site
polaxe   local axis type for each multipole site

MUTANT   **hybrid atoms for free energy perturbation**

lambda   weighting of initial state in hybrid Hamiltonian
nhybrid   number of atoms mutated from initial to final state
ihybrid   atomic sites differing in initial and final state
type0   atom type of each atom in the initial state system
class0   atom class of each atom in the initial state system
type1   atom type of each atom in the final state system
class1   atom class of each atom in the final state system
alter   true if an atom is to be mutated, false otherwise

OMEGA   **dihedrals for torsional space computations**

dihed   current value in radians of each dihedral angle
nomega   number of dihedral angles allowed to rotate
iomega   numbers of two atoms defining rotation axis
zline   line number in Z-matrix of each dihedral angle

OPBEND   **out-of-plane bends in the current structure**

kopb   force constant values for out-of-plane bending
nopbend   total number of out-of-plane bends in the system
iopb   bond angle numbers used in out-of-plane bending

OUTPUT   **control of coordinate output file format**

archive   logical flag to save structures in an archive
binary   logical flag to substitute binary for ASCII format
noversion   logical flag governing use of filename versions
overwrite   logical flag to overwrite intermediate files inplace
savecycle   logical flag to mark use of numbered cycle files
coordtype   selects Cartesian, internal, rigid body or none

PATHS   **parameters for Elber reaction path method**

p0   reactant Cartesian coordinates as variables
p1   product Cartesian coordinates as variables
pmid   midpoint between the reactant and product
pvect   vector connecting the reactant and product
pstep   step per cycle along reactant-product vector
pzet   current projection on reactant-product vector
pnorm   length of the reactant-product vector
acoeff   transformation matrix 'A' from Elber paper
gc   gradients of the path constraints

PDB   **definition of a Protein Data Bank structure**

xpdb   x-coordinate of each atom stored in PDB format
ypdb   y-coordinate of each atom stored in PDB format
zpdb   z-coordinate of each atom stored in PDB format
npdb   number of atoms stored in Protein Data Bank format
resnum   number of the residue to which each atom belongs
npdb12   number of atoms directly bonded to each CONECT atom
ipdb12   atom numbers of atoms connected to each CONECT atom
pdblist   list of the Protein Data Bank atom number of each atom
pdbtyp   Protein Data Bank record type assigned to each atom
atmnam   Protein Data Bank atom name assigned to each atom
resnam   Protein Data Bank residue name assigned to each atom

PHIPSI   **phi-psi-omega-chi angles for a protein**

phi   value of the phi angle for each amino acid residue
psi   value of the psi angle for each amino acid residue
omega   value of the omega angle for each amino acid residue
chi   values of the chi angles for each amino acid residue
chiral   chirality of each amino acid residue (1=L, -1=D)
disulf   residue joined to each residue via a disulfide link

PIBOND   **bond orders for a conjugated pi-system**

pbpl   pi-bond orders for bonds in "planar" pi-system
pnpl   pi-bond orders for bonds in "nonplanar" pi-system

PICALC   **orbital energies for conjugated pi-system**

q   number of pi-electrons contributed by each atom
w   ionization potential of each pi-system atom
em   repulsion integral for each pi-system atom
nfill   number of filled pi-system molecular orbitals

PISTUF   **conjugated system in the current structure**

norbit   total number of pisystem orbitals in the system
iorbit   numbers of the atoms containing pisystem orbitals

piperp	atoms defining a normal plane to each orbital
npibond	total number of bonds affected by the pisystem
pibond	bond and piatom numbers for each pisystem bond
npitors	total number of torsions affected by the pisystem
pitors	torsion and pibond numbers for each pisystem torsion
listpi	atom list indicating whether each atom has an orbital

PITERM   **bonds and torsions in the current pi-system**

bkpi	bond stretch force constants for pi-bond order of 1.0
blpi	ideal bond length values for a pi-bond order of 1.0
kslope	rate of force constant decrease with bond order decrease
lslope	rate of bond length increase with a bond order decrease
torspi	2-fold torsional energy barrier for pi-bond order of 1.0

PME   **parameters for particle mesh Ewald summation**

maxfft	maximum number of points along each FFT direction
maxorder	maximum order of the B-spline approximation
maxtable	maximum size of the FFT table array
maxgrid	maximum dimension of the PME charge grid array
bsmod1	B-spline moduli along the a-axis direction
bsmod2	B-spline moduli along the b-axis direction
bsmod3	B-spline moduli along the c-axis direction
table	intermediate array used by the FFT calculation
aewald	Ewald convergence coefficient value (Ang-1)
nfft1	number of grid points along the a-axis direction
nfft2	number of grid points along the b-axis direction
nfft3	number of grid points along the c-axis direction
order	order of the PME B-spline approximation

POLAR   **polarizabilities and induced dipole moments**

polarize	dipole polarizability for each multipole site (Ang**3)
pdamp	value of polarizability damping factor for each site
uind	induced dipole components at each multipole site
npolar	total number of polarizable sites in the system

POLPOT   **specifics of polarization functional form**

poleps	induced dipole convergence criterion (rms Debyes/atom)
pradius	radius of an idealized atom with unit polarizability
pgamma	prefactor in exponential polarization damping term
poltyp	type of polarization potential (direct or mutual)

POTENT   **usage of each potential energy component**

use_bond	logical flag governing use of bond stretch potential
use_angle	logical flag governing use of angle bend potential
use_strbnd	logical flag governing use of stretch-bend potential
use_urey	logical flag governing use of Urey-Bradley potential
use_angang	logical flag governing use of angle-angle cross term

use_opbend	logical flag governing use of out-of-plane bend term
use_improp	logical flag governing use of improper dihedral term
use_imptor	logical flag governing use of improper torsion term
use_tors	logical flag governing use of torsional potential
use_strtor	logical flag governing use of stretch-torsion term
use_tortor	logical flag governing use of torsion-torsion term
use_vdw	logical flag governing use of vdw der Waals potential
use_charge	logical flag governing use of charge-charge potential
use_chgdpl	logical flag governing use of charge-dipole potential
use_dipole	logical flag governing use of dipole-dipole potential
use_mpole	logical flag governing use of multipole potential
use_polar	logical flag governing use of polarization term
use_ewald	logical flag governing use of Ewald summation term
use_rxnfld	logical flag governing use of reaction field term
use_solv	logical flag governing use of surface area solvation
use_gbsa	logical flag governing use of GB/SA solvation term
use_geom	logical flag governing use of geometric restraints
use_extra	logical flag governing use of extra potential term
use_orbit	logical flag governing use of pi-system computation

PRECIS   **values of machine precision tolerances**

tiny	the smallest positive floating point value
small	the smallest relative floating point spacing
huge	the largest relative floating point spacing

REFER   **storage of reference atomic coordinate set**

xref	reference x-coordinate for each atom in the system
yref	reference y-coordinate for each atom in the system
zref	reference z-coordinate for each atom in the system
nref	total number of atoms in the reference system
reftyp	atom type for each atom in the reference system
n12ref	number of atoms bonded to each reference atom
i12ref	atom numbers of atoms 1-2 connected to each atom
refleng	length in characters of the reference filename
refltitle	length in characters of the reference title string
refnam	atom name for each atom in the reference system
reffile	base filename for the reference structure
reftitle	title used to describe the reference structure

RESTRN   **definition of the geometrical restraints**

depth	depth of shallow Gaussian basin restraint
width	exponential width coefficient of Gaussian basin
rwall	radius of spherical droplet boundary restraint
xpfix	x-coordinate target for each restrained position
ypfix	y-coordinate target for each restrained position
zpfix	z-coordinate target for each restrained position
pfix	flat-well range and force constant for each position
dfix	target range and force constant for each distance
tfix	target range and force constant for each torsion

npfix	number of position restraints to be applied
ipfix	atom number involved in each position restraint
ndfix	number of distance restraints to be applied
idfix	atom numbers defining each distance restraint
ntfix	number of torsional restraints to be applied
itfix	atom numbers defining each torsional restraint
use_basin	logical flag governing use of Gaussian basin
use_wall	logical flag governing use of droplet boundary

RIGID　　　　　　　　　　**rigid body coordinates for atom groups**

xrb	rigid body reference x-coordinate for each atom
yrb	rigid body reference y-coordinate for each atom
zrb	rigid body reference z-coordinate for each atom
rbc	current rigid body coordinates for each atom group

RING　　　　　　　　　　**number and location of small ring structures**

nring3	total number of 3-membered rings in the system
iring3	numbers of the atoms involved in each 3-ring
nring4	total number of 4-membered rings in the system
iring4	numbers of the atoms involved in each 4-ring
nring5	total number of 5-membered rings in the system
iring5	numbers of the atoms involved in each 5-ring
nring6	total number of 6-membered rings in the system
iring6	numbers of the atoms involved in each 6-ring

ROTATE　　　　　　　　　　**molecule partitions for rotation of a bond**

nrot	total number of atoms moving when bond rotates
rot	atom numbers of atoms moving when bond rotates
use_short	logical flag to enforce use of shortest atom list

RXNFLD　　　　　　　　　　**reaction field matrix elements and indices**

b1	
b2	
ijk	

RXNPOT　　　　　　　　　　**specifics of reaction field functional form**

rfsize	radius of reaction field sphere centered at origin
rfbulkd	bulk dielectric constant of reaction field continuum
rfterms	number of terms to use in reaction field summation

SCALES　　　　　　　　　　**parameter scale factors for optimization**

scale	multiplicative factor for each optimization parameter
set_scale	logical flag to show if scale factors have been set

SEQUEN　　　　　　　　　　**sequence information for a biopolymer**

nseq	total number of residues in biopolymer sequences
nchain	number of separate biopolymer sequence chains
ichain	first and last residue in each biopolymer chain
seqtyp	residue type for each residue in the sequence
seq	one-letter code for each residue in the sequence
chnnam	one-letter identifier for each sequence chain

SHAKE   **definition of Shake/Rattle constraints**

krat	ideal distance value for rattle constraint
nrat	number of rattle constraints to be applied
irat	atom numbers of atoms in a rattle constraint
ratimage	flag to use minimum image for rattle constraint
use_rattle	logical flag to set use of rattle contraints

SHUNT   **polynomial switching function coefficients**

off	distance at which the potential energy goes to zero
off2	square of distance at which the potential goes to zero
cut	distance at which switching of the potential begins
cut2	square of distance at which the switching begins
c0	zeroth order coefficient of multiplicative switch
c1	first order coefficient of multiplicative switch
c2	second order coefficient of multiplicative switch
c3	third order coefficient of multiplicative switch
c4	fourth order coefficient of multiplicative switch
c5	fifth order coefficient of multiplicative switch
f0	zeroth order coefficient of additive switch function
f1	first order coefficient of additive switch function
f2	second order coefficient of additive switch function
f3	third order coefficient of additive switch function
f4	fourth order coefficient of additive switch function
f5	fifth order coefficient of additive switch function
f6	sixth order coefficient of additive switch function
f7	seventh order coefficient of additive switch function

SIZES   **parameter values to set array dimensions**

"sizes.i" sets values for critical array dimensions used throughout the software; these parameters will fix the size of the largest systems that can be handled; values too large for the computer's memory and/or swap space to accomodate will result in poor performance or outright failure

| parameter: | maximum allowed number of: |

maxatm	atoms in the molecular system
maxval	atoms directly bonded to an atom
maxgrp	user-defined groups of atoms
maxtyp	force field atom type definitions
maxclass	force field atom class definitions
maxkey	lines in the keyword file
maxrot	bonds for torsional rotation

maxhess	off-diagonal Hessian elements
maxopt	optimization variables (full-matrix)
maxvib	vibrational frequencies
maxgeo	distance geometry points
maxpi	atoms in conjugated pisystem
maxcell	unit cells in replicated crystal
maxring	3-, 4-, or 5-membered rings
maxfix	geometric restraints
maxbio	biopolymer atom definitions
maxres	residues in the macromolecule
maxamino	amino acid residue types
maxvar	optimization variables (linear storage)
maxbnd	covalent bonds in molecular system
maxang	bond angles in molecular system
maxtors	dihedral angles in molecular system
maxlight	sites for method of lights neighbors
maxpib	covalent bonds in pisystem
maxpit	dihedrals involving pisystem

SOLUTE   **surface area-based macroscopic solvation**

rsolv	atomic radius of each atom for empirical solvation
vsolv	atomic solvation parameters (kcal/mole/Ang**2)
rborn	Born radius of each atom for GB/SA solvation
nsolv	number of atoms with non-zero solvation parameters
reborn	number of GB/SA calculations between Born radii updates
bornmax	maximum atoms for original Born radii computation

STODYN   **frictional coefficients for SD trajectory**

friction	global frictional coefficient for exposed particle
gamma	atomic frictional coefficients for each atom

STRBND   **stretch-bends in the current structure**

ksb	force constant for stretch-bend terms
nstrbnd	total number of stretch-bend interactions
isb	angle and bond numbers used in stretch-bend

STRTOR   **stretch-torsions in the current structure**

kst	1-, 2- and 3-fold stretch-torsion force constants
nstrtor	total number of stretch-torsion interactions
ist	torsion and bond numbers used in stretch-torsion

SYNTRN   **definition of synchronous transit path**

t	value of the path coordinate (0=reactant, 1=product)
pm	path coordinate for extra point in quadratic transit
xmin1	reactant coordinates as array of optimization variables
xmin2	product coordinates as array of optimization variables
xm	extra coordinate set for quadratic synchronous transit

TITLES　　　　　　　　　**title for the current molecular system**

ltitle　　　　　　　　　length in characters of the nonblank title string
title　　　　　　　　　title used to describe the current structure

TORPOT　　　　　　　　　**specifics of torsional functional forms**

torsunit　　　　　　　　　scale factor for torsional parameter amplitudes
storunit　　　　　　　　　convert stretch-torsion force to kcal/mole/Ang

TORS　　　　　　　　　**torsional angles within the current structure**

tors1　　　　　　　　　1-fold amplitude and phase for each torsional angle
tors2　　　　　　　　　2-fold amplitude and phase for each torsional angle
tors3　　　　　　　　　3-fold amplitude and phase for each torsional angle
tors4　　　　　　　　　4-fold amplitude and phase for each torsional angle
tors5　　　　　　　　　5-fold amplitude and phase for each torsional angle
tors6　　　　　　　　　6-fold amplitude and phase for each torsional angle
ntors　　　　　　　　　total number of torsional angles in the system
itors　　　　　　　　　numbers of the atoms in each torsional angle

TREE　　　　　　　　　**potential smoothing & search tree levels**

maxpss　　　　　　　　　maximum number of potential smoothing levels
etree　　　　　　　　　energy reference value at the top of the tree
ilevel　　　　　　　　　smoothing deformation value at each tree level
nlevel　　　　　　　　　number of levels of potential smoothing used

UNITS　　　　　　　　　**physical constants and unit conversions**

avogadro　　　　　　　　　Avogadro's number (N) in particles/mole
boltzmann　　　　　　　　　Boltzmann constant (kB) in g*Ang**2/ps**2/K/mole
gasconst　　　　　　　　　ideal gas constant (R) in kcal/mole/K
lightspd　　　　　　　　　speed of light in vacuum (c) in cm/ps
bohr　　　　　　　　　conversion from Bohrs to Angstroms
joule　　　　　　　　　conversion from calories to joules
evolt　　　　　　　　　conversion from Hartree to electron-volts
hartree　　　　　　　　　conversion from Hartree to kcal/mole
electric　　　　　　　　　conversion from electron**2/Ang to kcal/mole
debye　　　　　　　　　conversion from electron-Ang to Debyes
prescon　　　　　　　　　conversion from kcal/mole/Ang**3 to Atm
convert　　　　　　　　　conversion from kcal to g*Ang**2/ps**2

UREY　　　　　　　　　**Urey-Bradley interactions in the structure**

uk　　　　　　　　　Urey-Bradley force constants (kcal/mole/Ang**2)
ul　　　　　　　　　ideal 1-3 distance values in Angstroms
nurey　　　　　　　　　total number of Urey-Bradley terms in the system
iury　　　　　　　　　numbers of the atoms in each Urey-Bradley interaction

URYPOT　　　　　　　　　**specifics of Urey-Bradley functional form**

ureyunit	convert Urey-Bradley constant to kcal/mole/Ang**2

USAGE   **atoms active during energy computation**

nuse	number of active atoms used in energy calculation
use	true if an atom is active, false if inactive

VDW   **van der Waals parameters for current structure**

radmin	minimum energy distance for each atom class pair
epsilon	well depth parameter for each atom class pair
radmin4	minimum energy distance for 1-4 interaction pairs
epsilon4	well depth parameter for 1-4 interaction pairs
radhbnd	minimum energy distance for hydrogen bonding pairs
epshbnd	well depth parameter for hydrogen bonding pairs
kred	value of reduction factor parameter for each atom
ired	attached atom from which reduction factor is applied
nvdw	total number van der Waals active sites in the system
ivdw	number of the atom for each van der Waals active site

VDWPOT   **specifics of van der Waals functional form**

aterm	value of the "A" constant in exp-6 vdw potential
bterm	value of the "B" constant in exp-6 vdw potential
cterm	value of the "C" constant in exp-6 vdw potential
vdwscale	factor by which 1-4 vdw interactions are scaled
igauss	coefficients of Gaussian fit to vdw potential
ngauss	number of Gaussians used in fit to vdw potential
vdw12use	usage of 1-2 vdw terms (1=omit, -1=use scaled value)
vdw13use	usage of 1-3 vdw terms (1=omit, -1=use scaled value)
vdw14use	usage of 1-4 vdw terms (1=omit, -1=use scaled value)
vdwtyp	type of van der Waals potential energy function
radtyp	type of parameter (sigma or R-min) for atomic size
radsiz	atomic size provided as radius or diameter
radrule	combining rule for atomic size parameters
epsrule	combining rule for vdw well depth parameters
gausstyp	type of Gaussian fit to van der Waals potential

VIRIAL   **components of the internal virial**

virx	x-component of the total internal virial
viry	y-component of the total internal virial
virz	z-component of the total internal virial

WARP   **parameters for potential surface smoothing**

m2	second moment of Gaussian representing each atom
deform	value of diffusional smoothing deformation parameter
diffb	diffusion coefficient for bond stretch potential
diffa	diffusion coefficient for angle bend potential
diffid	diffusion coefficient for improper dihedral potential

difft	diffusion coefficient for torsional potential
diffv	diffusion coefficient for van der Waals potential
diffc	diffusion coefficient for charge-charge potential
use_deform	flag to use diffusion smoothed potential terms
use_gda	flag to use Straub's GDA instead of Scheraga's DEM

XTALS   **crystal structures for parameter fitting**

e0_lattice	ideal lattice energy for the current crystal
moment_0	ideal dipole moment for monomer from crystal
nxtal	number of crystal structures to be stored
nvary	number of potential parameters to optimize
ivary	index for the types of potential parameters
vary	atom numbers involved in potential parameters
iresid	crystal structure to which each residual refers
rsdtyp	experimental variable for each of the residuals
vartyp	type of potential parameter to be optimized

ZCLOSE   **ring openings and closures for Z-matrix**

nadd	number of added bonds between Z-matrix atoms
iadd	numbers of the atom pairs defining added bonds
ndel	number of bonds between Z-matrix bonds to delete
idel	numbers of the atom pairs defining deleted bonds

ZCOORD   **Z-matrix internal coordinate definitions**

zbond	bond length used to define each Z-matrix atom
zang	bond angle used to define each Z-matrix atom
ztors	angle or torsion used to define Z-matrix atom
iz	defining atom numbers for each Z-matrix atom

11. Index of Function & Subroutine Calls

This section contains an alphabetical cross index listing of the routines called by each TINKER program, subroutine and function. Routines not present in the left hand column do not make calls to any other portion of the TINKER package.

Routine	List of Code Units called by this Routine			
ACTIVE	GETTEXT	UPCASE		
ADDSIDE	ADDBOND	FINDATM	NEWATM	OLDATM
ALCHEMY	ENERGY	FINAL	FREEUNIT	GETTEXT	GETXYZ
	HATOM	HYBRID	INITIAL	MECHANIC	NUMERAL
	READXYZ	UPCASE	VERSION		
ANALYSIS	BOUNDS	EANGANG3	EANGLE3	EBOND3	EBUCK3
	ECHARGE3	ECHARGE9	ECHGDPL3	EDIPOLE3	EGAUSS3
	EGEOM3	EHAL3	EIMPROP3	EIMPTOR3	ELJ3
	EMM3HB3	EMPOLE3	EOPBEND3	ERXNFLD3	ESOLV3
	ESTRBND3	ESTRTOR3	ETORS3	EUREY3	EWALD3
	EXTRA3	PISCF			
ANALYZE	ANALYSIS	FINAL	GETXYZ	GYRATE	INERTIA
	INITIAL	MECHANIC	NEXTARG	TRIMTEXT	UPCASE
ANGLES	FATAL				
ANNEAL	BEEMAN	BORN	FINAL	GETTEXT	GETWORD
	GETXYZ	INITIAL	MDINIT	MDREST	MECHANIC
	NEXTARG	SHAKEUP	SIGMOID	UPCASE	VERLET
ARCHIVE	ACTIVE	BASEFILE	FINAL	FREEUNIT	GETTEXT
	INITIAL	NEXTARG	NUMERAL	PRTBIOS	PRTXMOL
	PRTXYZ	READXYZ	SUFFIX	TRIMTEXT	UPCASE
	VERSION				
ATTACH	SORT				
BASEFILE	CONTROL	GETKEY	TRIMTEXT		
BEEMAN	GRADIENT	MDSTAT	PRESSURE	RATTLE	RATTLE2
	TEMPER				
BETAI	BETACF	GAMMLN			
BONDS	FATAL				
BORN	SURFATOM				
BSET	BMAX				

BSSTEP	DERIVS	FATAL	MMID	PZEXTR
CALENDAR	ITIME			
CFFTB	CFFTB1			
CFFTB1	PASSB	PASSB2	PASSB3	PASSB4	PASSB5
CFFTF	CFFTF1			
CFFTF1	PASSF	PASSF2	PASSF3	PASSF4	PASSF5
CFFTI	CFFTI1			
CHKTREE	PSSMIN			
CIRPLN	ANORM	DOT	VCROSS	VNORM
CLIMBER	ENERGY	GETREF	LOCALMIN	MAKEINT	MAKEXYZ
CLIMBRGD	ENERGY	LOCALRGD	RIGIDXYZ	
CLIMBROT	ENERGY	LOCALROT	MAKEXYZ	
CLIMBROT	CHKTREE PSSMIN	ENERGY	GETREF	MAKEINT	MAKEXYZ
CLIMBXYZ	CHKTREE	ENERGY	GETREF	PSSMIN
CLOCK	ETIME			
CLUSTER	FATAL UPCASE	GETNUMB	GETTEXT	SORT	SORT3
COMMAND	GETARG	IARGC	UPCASE	
COMPRESS	ERROR	GETTOR		
CONNECT	SORT			
CONNOLLY	COMPRESS TORUS	CONTACT VAM	NEIGHBOR	PLACE	SADDLES
CONTACT	ANORM	ERROR	PTINCY	
CONTROL	GETTEXT	UPCASE		
COORDS	GYRATE	RMSERROR		
CORRELATE	FINAL	INITIAL	NEXTARG	PROPERTY	READBLK

TRIMTEXT

CRYSTAL	BOUNDS	FIELD	FINAL	FREEUNIT	GETTEXT
	GETXYZ	INITIAL	KATOM	LATTICE	MAKE27
	MOLECULE	PRTXYZ	SYMMETRY	UNITCELL	UPCASE
	VERSION				
CUTOFFS	GETTEXT	UPCASE			
DEPTH	DOT	VCROSS	VNORM		
DIAGQ	GETIME	SETIME			
DIFFEQ	BSSTEP	DERIVS	GDASTAT		
DISTGEOM	ACTIVE	ANGLES	ATTACH	BONDS	CHIRIN
	EMBED	FATAL	FINAL	FREEUNIT	GEODESIC
	GETIME	GETTEXT	GETXYZ	GRAFIC	IMPOSE
	INITIAL	MAKEREF	NEXTARG	NUMERAL	PRTXYZ
	RESTRAIN	SETIME	TORSIONS	TRIFIX	UPCASE
	VERSION				
DMDUMP	GRAFIC				
DOCUMENT	FINAL	FREEUNIT	GETPRM	GETTEXT	GETWORD
	INITIAL	LOWCASE	NEXTARG	NEXTTEXT	PRTPRM
	SORT6	SORT7	SUFFIX	TRIMTEXT	UPCASE
	VERSION				
DROTMAT	DROTMAT1	DROTMAT2			
DSTMAT	GETIME	GETNUMB	GETTEXT	INVBETA	LOWCASE
	RANDOM	SETIME	SORT2	TRIFIX	UPCASE
DYNAMIC	BEEMAN	BORN	FINAL	GETTEXT	GETWORD
	GETXYZ	INITIAL	MDINIT	MDREST	MECHANIC
	NEXTARG	SDSTEP	SHAKEUP	UPCASE	VERLET
EANGANG	GROUPS				
EANGANG1	GROUPS				
EANGANG2	EANGANG2B	GROUPS			
EANGANG3	GROUPS				
EANGLE	GROUPS				
EANGLE1	GROUPS				
EANGLE2	EANGLE2A	EANGLE2B	GROUPS		

EANGLE2A	GROUPS		
EANGLE3	GROUPS		
EBOND	GROUPS		
EBOND1	GROUPS		
EBOND2	GROUPS		
EBOND3	GROUPS		
EBUCK	GROUPS	IMAGE	SWITCH
EBUCK1	GROUPS	IMAGE	SWITCH
EBUCK2	GROUPS	IMAGE	SWITCH
EBUCK3	GROUPS	IMAGE	SWITCH
EBUCK4	GROUPS	LIGHTS	SWITCH
EBUCK5	GROUPS	LIGHTS	SWITCH
ECHARGE	GROUPS	IMAGE	SWITCH
ECHARGE1	GROUPS	IMAGE	SWITCH
ECHARGE2	GROUPS	IMAGE	SWITCH
ECHARGE3	GROUPS	IMAGE	SWITCH
ECHARGE4	GROUPS	LIGHTS	SWITCH
ECHARGE5	GROUPS	LIGHTS	SWITCH
ECHARGE6	ERF	GROUPS	SWITCH
ECHARGE7	ERF	GROUPS	SWITCH
ECHARGE8	ERF	GROUPS	SWITCH
ECHARGE9	ERF	GROUPS	SWITCH
ECHGDPL	GROUPS	IMAGE	SWITCH
ECHGDPL1	GROUPS	IMAGE	SWITCH
ECHGDPL2	GROUPS	IMAGE	SWITCH
ECHGDPL3	GROUPS	IMAGE	SWITCH

EDIPOLE	GROUPS	IMAGE	SWITCH
EDIPOLE1	GROUPS	IMAGE	SWITCH
EDIPOLE2	GROUPS	IMAGE	SWITCH
EDIPOLE3	GROUPS	IMAGE	SWITCH
EGAUSS	GROUPS	SWITCH	
EGAUSS1	GROUPS	SWITCH	
EGAUSS2	GROUPS	SWITCH	
EGAUSS3	GROUPS	SWITCH	
EGBSA	IMAGE	SWITCH	
EGBSA1	IMAGE	SWITCH	
EGBSA2	IMAGE	SWITCH	
EGBSA3	IMAGE	SWITCH	
EGEOM	IMAGE		
EGEOM1	IMAGE		
EGEOM2	IMAGE		
EGEOM3	IMAGE		
EHAL	GROUPS	IMAGE	SWITCH
EHAL1	GROUPS	IMAGE	SWITCH
EHAL2	GROUPS	IMAGE	SWITCH
EHAL3	GROUPS	IMAGE	SWITCH
EHAL4	GROUPS	LIGHTS	SWITCH
EHAL5	GROUPS	LIGHTS	SWITCH
EIGEN	GETIME	POWER	SETIME
EIGENRGD	DIAGQ	HESSRGD	
EIGENROT	DIAGQ	HESSROT	

EIGENROT	DIAGQ	HESSROT		
EIGENROT	DIAGQ	HESSROT		
EIGENXYZ	DIAGQ	HESSIAN		
EIMPROP	GROUPS			
EIMPROP1	GROUPS			
EIMPROP2	GROUPS			
EIMPROP3	GROUPS			
EIMPTOR	GROUPS			
EIMPTOR1	GROUPS			
EIMPTOR2	GROUPS			
EIMPTOR3	GROUPS			
ELJ	GROUPS	IMAGE	SWITCH	
ELJ1	GROUPS	IMAGE	SWITCH	
ELJ2	GROUPS	IMAGE	SWITCH	
ELJ3	GROUPS	IMAGE	SWITCH	
ELJ4	GROUPS	LIGHTS	SWITCH	
ELJ5	GROUPS	LIGHTS	SWITCH	
EMBED	BNDERR	CHIRER	CHKSIZE	COORDS	DMDUMP
	DSTMAT	EIGEN	EXPLORE	FRACDIST	FREEUNIT
	GETIME	GYRATE	IMPOSE	LOCERR	MAJORIZE
	METRIC	NUMERAL	PRTXYZ	REFINE	RMSERROR
	SETIME	TORSER	VDWERR		
EMM3HB	GROUPS	IMAGE	SWITCH	
EMM3HB1	GROUPS	IMAGE	SWITCH	
EMM3HB2	GROUPS	IMAGE	SWITCH	
EMM3HB3	GROUPS	IMAGE	SWITCH	
EMM3HB4	GROUPS	LIGHTS	SWITCH	
EMM3HB5	GROUPS	LIGHTS	SWITCH	

EMPOLE	EMPIK	GROUPS	IMAGE	INDUCE	ROTMAT
	ROTPOLE	SWITCH			
EMPOLE1	DROTMAT	DROTPOLE	EMPIK1	GROUPS	IMAGE
	INDUCE	ROTMAT	ROTPOLE	SWITCH	
EMPOLE2	EMPOLE2B				
EMPOLE2B	DROTMAT	DROTPOLE	EMPIK1	GROUPS	IMAGE
	INDUCE	ROTMAT	ROTPOLE	SWITCH	
EMPOLE3	EMPIK	GROUPS	IMAGE	INDUCE	ROTMAT
	ROTPOLE	SWITCH			
ENERGY	BOUNDS	EANGANG	EANGLE	EBOND	EBUCK
	EBUCK4	ECHARGE	ECHARGE4	ECHARGE6	ECHGDPL
	EDIPOLE	EGAUSS	EGEOM	EHAL	EHAL4
	EIMPROP	EIMPTOR	ELJ	ELJ4	EMM3HB
	EMM3HB4	EMPOLE	EOPBEND	ERXNFLD	ESOLV
	ESTRBND	ESTRTOR	ETORS	EUREY	EWALD
	EXTRA	PISCF			
EOPBEND	GROUPS				
EOPBEND1	GROUPS				
EOPBEND2	EOPBEND2B	GROUPS			
EOPBEND3	GROUPS				
EPME	BSPLINE	FFTFRONT			
EPME1	BSPLINE	FFTBACK	FFTFRONT		
EPUCLC	ANORM				
ERF	ERFCORE				
ERFC	ERFCORE				
ERFIK	D1D2	RFINDEX			
ERFINV	ERF	FATAL			
ERROR	FATAL	TRIMTEXT			
ERXNFLD	ERFIK	IJK_PT	ROTMAT	ROTPOLE	SWITCH
ERXNFLD3	ERFIK	IJK_PT	ROTMAT	ROTPOLE	SWITCH
ESOLV	BORN	EGBSA	SURFACE		

ESOLV1	BORN	EGBSA1	SURFACE	
ESOLV2	EGBSA2			
ESOLV3	BORN	EGBSA3	SURFACE	
ESTRBND	GROUPS			
ESTRBND1	GROUPS			
ESTRBND2	GROUPS			
ESTRBND3	GROUPS			
ESTRTOR	GROUPS			
ESTRTOR1	GROUPS			
ESTRTOR2	GROUPS			
ESTRTOR3	GROUPS			
ETORS	GROUPS			
ETORS1	GROUPS			
ETORS2	GROUPS			
ETORS3	GROUPS			
EUREY	GROUPS			
EUREY1	GROUPS			
EUREY2	GROUPS			
EUREY3	GROUPS			
EWALD	EPME	ERFC	GROUPS	IMAGE
EWALD1	EPME1	ERFC	GROUPS	IMAGE
EWALD2	ERFC	GROUPS	IMAGE	
EWALD3	EPME	ERFC	GROUPS	IMAGE
EWALDCOF	ERFC			
EXPLORE	INITER	MIDERR	SIGMOID	TOTERR

FFTBACK	CFFTB				
FFTFRONT	CFFTF				
FFTSETUP	CFFTI				
FIELD	GETPRM	PRMKEY			
FRACDIST	TRIMTEXT				
FREEUNIT	FATAL				
GDA	DIFFEQ	FINAL	FREEUNIT	GDA1	GDA2
	GDA3	GDASTAT	GETTEXT	GETXYZ	INITIAL
	MECHANIC	NEXTARG	NUMERAL	PRTXYZ	RANDOM
	TNCG	UPCASE	VERSION	WRITEOUT	
GDA1	GRADIENT	HESSIAN			
GDA2	GRADIENT				
GDA3	HESSIAN				
GDASTAT	ENERGY	GYRATE	WRITEOUT		
GEODESIC	MINPATH	SORT3			
GETHYDRO	PDBATM				
GETIME	CLOCK				
GETINT	BASEFILE	CONNECT	FREEUNIT	MAKEXYZ	NEXTARG
	READINT	SUFFIX	VERSION		
GETKEY	FATAL	FREEUNIT	GETTEXT	SUFFIX	TRIMTEXT
	UPCASE				
GETMOL2	BASEFILE	FREEUNIT	NEXTARG	READMOL2	SUFFIX
	VERSION				
GETNUMB	TRIMTEXT				
GETPDB	BASEFILE	FREEUNIT	NEXTARG	READPDB	SUFFIX
	VERSION				
GETPRB	DIST2	DOT	GETTOR	VCROSS	
GETPRM	FREEUNIT	GETTEXT	INITPRM	NEXTARG	READPRM
	SUFFIX	UPCASE	VERSION		
GETSEQ	GETWORD	TRIMTEXT	UPCASE		

GETSIDE	PDBATM			
GETTOR	DIST2			
GETXYZ	BASEFILE	FREEUNIT	NEXTARG	READXYZ	SUFFIX
	VERSION				
GRADIENT	BOUNDS	EANGANG1	EANGLE1	EBOND1	EBUCK1
	EBUCK5	ECHARGE1	ECHARGE5	ECHARGE7	ECHGDPL1
	EDIPOLE1	EGAUSS1	EGEOM1	EHAL1	EHAL5
	EIMPROP1	EIMPTOR1	ELJ1	ELJ5	EMM3HB1
	EMM3HB5	EMPOLE1	EOPBEND1	ERXNFLD1	ESOLV1
	ESTRBND1	ESTRTOR1	ETORS1	EUREY1	EWALD1
	EXTRA1	PISCF			
GRADRGD	GRADIENT				
GRADROT	GRADIENT	ROTLIST			
HANGLE	NUMERAL				
HBOND	NUMERAL				
HDIPOLE	NUMERAL				
HESSIAN	BORN	BOUNDS	EANGANG2	EANGLE2	EBOND2
	EBUCK2	ECHARGE2	ECHARGE8	ECHGDPL2	EDIPOLE2
	EGAUSS2	EGEOM2	EHAL2	EIMPROP2	EIMPTOR2
	ELJ2	EMM3HB2	EMPOLE2	EOPBEND2	ERXNFLD2
	ESOLV2	ESTRBND2	ESTRTOR2	ETORS2	EUREY2
	EWALD2	EXTRA2	FATAL	PISCF	
HESSRGD	GRADRGD	RIGIDXYZ			
HESSROT	GRADROT	MAKEXYZ			
HIMPTOR	NUMERAL				
HSTRTOR	NUMERAL				
HTORS	NUMERAL				
HYBRID	HANGLE	HATOM	HBOND	HCHARGE	HDIPOLE
	HIMPTOR	HSTRBND	HSTRTOR	HTORS	HVDW
IMPOSE	CENTER	QUATFIT	RMSFIT		
INDUCE	FATAL	IMAGE	PRTERR	SWITCH	UDIRECT
	UMUTUAL				
INEDGE	ERROR				

INERTIA	JACOBI			
INITER	LOCERR	TORSER		
INITIAL	COMMAND	INITRES	PRECISE	PROMO
INITROT	FATAL	NEXTARG	ROTCHECK	ROTLIST
INTEDIT	BNDANGLE GETINT TRIMTEXT	DIHEDRAL GETWORD UPCASE	FIELD INITIAL VERSION	FINAL MAKEXYZ ZHELP	FREEUNIT PRTINT ZVALUE
INTXYZ	FINAL VERSION	FREEUNIT	GETINT	INITIAL	PRTXYZ
INVBETA	BETAI	GAMMLN		
INVERT	FATAL			
IPEDGE	ERROR			
KANGANG	GETTEXT	UPCASE		
KANGLE	GETTEXT	NUMERAL	UPCASE	
KATOM	GETNUMB	GETSTRING	GETTEXT	UPCASE
KBOND	GETTEXT	NUMERAL	UPCASE	
KCHARGE	GETTEXT	UPCASE		
KDIPOLE	GETTEXT	NUMERAL	UPCASE	
KEWALD	EWALDCOF UPCASE	FATAL	FFTSETUP	GETTEXT	MODULI
KIMPROP	GETTEXT	NUMERAL	UPCASE	
KIMPTOR	GETTEXT	NUMERAL	TORPHASE	UPCASE
KMPOLE	GETTEXT	NUMBER	NUMERAL	SORT3	UPCASE
KOPBEND	GETTEXT	NUMBER	NUMERAL	UPCASE
KORBIT	GETTEXT	NUMERAL	UPCASE	
KPOLAR	GETTEXT	UPCASE		
KSTRBND	GETTEXT	UPCASE		
KSTRTOR	GETTEXT	NUMERAL	UPCASE	

KTORS	GETTEXT	NUMERAL	TORPHASE	UPCASE	
KUREY	GETTEXT	NUMERAL	UPCASE		
KVDW	GETTEXT	NUMBER	NUMERAL	UPCASE	
LATTICE	FATAL				
LIGHTS	FATAL	SORT2	SORT5		
LMQN	BORN WRITEOUT	FGVALUE	GETTEXT	SEARCH	UPCASE
LMSTEP	PRECISE	QRSOLVE			
LOCALMIN	GRADIENT	SCAN1	SCAN2	TNCG	WRITEOUT
LOCALRGD	OCVM	PSSRGD1	WRITEOUT		
LOCALROT	OCVM	PSSROT1	WRITEOUT		
MAJORIZE	GETIME	GYRATE	RMSERROR	SETIME	
MAKE27	CELLATOM				
MAKEINT	ADJACENT UPCASE	BNDANGLE	DIHEDRAL	FATAL	GETTEXT
MAKEPDB	ATTACH PDBATM	FREEUNIT READSEQ	GETHYDRO VERSION	GETSIDE	NUMERAL
MAKEXYZ	XYZATM				
MAPCHECK	FREEUNIT	NUMERAL	PRTXYZ	VERSION	
MAXWELL	ERFINV	RANDOM			
MDINIT	FREEUNIT NUMERAL	GRADIENT RANVEC	LATTICE READDYN	MAXWELL VERSION	MDREST
MDREST	INVERT				
MDSTAT	FATAL SUFFIX	FREEUNIT VERSION	NUMERAL	PRTDYN	PRTXYZ
MEASFN	ERROR	TRIPLE	VCROSS	VECANG	VNORM
MEASFP	DOT	ERROR	VCROSS	VECANG	VNORM
MEASFS	DOT	ERROR	VECANG	VNORM	

MEASPM	VCROSS				
MECHANIC	ACTIVE	ANGLES	ATTACH	BONDS	CLUSTER
	CUTOFFS	FATAL	FIELD	KANGANG	KANGLE
	KATOM	KBOND	KCHARGE	KDIPOLE	KEWALD
	KIMPROP	KIMPTOR	KMPOLE	KOPBEND	KORBIT
	KPOLAR	KSTRBND	KSTRTOR	KTORS	KUREY
	KVDW	LATTICE	MOLECULE	MUTATE	ORBITAL
	RESTRAIN	RINGS	SMOOTH	SOLVATE	TORSIONS
	UNITCELL				
MERGE	FATAL	GETREF			
MIDERR	BNDERR	CHIRER	LOCERR	TORSER	
MINIMIZ1	GRADIENT				
MINIMIZE	FINAL	FREEUNIT	GETXYZ	GRADIENT	INITIAL
	LMQN	MECHANIC	MINIMIZ1	NEXTARG	PRTXYZ
	VERSION	WRITEOUT			
MINIROT	FINAL	FREEUNIT	GETINT	GRADROT	INITIAL
	INITROT	LMQN	MECHANIC	MINIROT1	NEXTARG
	PRTINT	VERSION	WRITEOUT		
MINIROT1	GRADROT	MAKEXYZ			
MMID	DERIVS				
MODECART	CLIMBXYZ	EIGENXYZ	GETREF	IMPOSE	MAKEREF
MODESRCH	CLIMBER	EIGENROT	MAKEINT	MAKEREF	MAPCHECK
MODETORS	CLIMBROT	EIGENROT	MAKEXYZ		
MODETORS	CLIMBROT	EIGENROT	GETREF	IMPOSE	MAKEINT
	MAKEREF				
MODULI	BSPLINE	DFTMOD			
MOLECULE	SORT	SORT3			
MUTATE	GETTEXT	UPCASE			
NEIGHBOR	DIST2	ERROR			
NEWATM	ADDBOND	XYZATM			
NEWTON	FINAL	FREEUNIT	GETTEXT	GETXYZ	GRADIENT
	INITIAL	MECHANIC	NEWTON1	NEWTON2	NEXTARG
	PRTXYZ	TNCG	UPCASE	VERSION	WRITEOUT

NEWTON1	GRADIENT			
NEWTON2	HESSIAN			
NEWTROT	FINAL	FREEUNIT	GETINT	GETTEXT	GRADROT
	INITIAL	INITROT	MECHANIC	NEWTROT1	NEWTROT2
	NEXTARG	PRTINT	TNCG	UPCASE	VERSION
	WRITEOUT				
NEWTROT1	GRADROT	MAKEXYZ			
NEWTROT2	HESSROT	MAKEXYZ			
NORMAL	RANDOM				
NUMBER	TRIMTEXT				
NUMGRAD	FVALUE				
OCVM	BORN	FGVALUE	GETTEXT	PRECISE	UPCASE
	WRITEOUT				
OLDATM	ADDBOND	FATAL			
OPTIMIZ1	GRADIENT				
OPTIMIZE	FINAL	FREEUNIT	GETXYZ	GRADIENT	INITIAL
	MECHANIC	NEXTARG	OCVM	OPTIMIZ1	PRTXYZ
	VERSION	WRITEOUT			
OPTIROT	FINAL	FREEUNIT	GETINT	GRADROT	INITIAL
	INITROT	MECHANIC	NEXTARG	OCVM	OPTIROT1
	PRTINT	VERSION	WRITEOUT		
OPTIROT1	GRADROT	MAKEXYZ			
OPTRIGID	FINAL	FREEUNIT	GETXYZ	GRADRGD	INITIAL
	MECHANIC	NEXTARG	OCVM	OPTRIGID1	ORIENT
	PRTXYZ	VERSION	WRITEOUT		
OPTRIGID1	GRADRGD	RIGIDXYZ			
ORBITAL	FATAL	GETTEXT	PIPLANE	UPCASE	
ORIENT	XYZRIGID				
OVERLAP	SLATER				
PATH	FINAL	GETXYZ	IMPOSE	INITIAL	INVERT
	LMQN	MECHANIC	NEXTARG	ORTHOG	PATH1
	POTNRG	WRITEOUT			

PATH1	POTNRG			
PATHPNT	OCVM	TRANSIT	WRITEOUT	
PATHSCAN	PATHPNT	SADDLE1	TANGENT	
PATHVAL	IMPOSE			
PDBXYZ	DELETE	FIELD	FINAL	FREEUNIT	GETNUMB
	GETPDB	INITIAL	PRTXYZ	RIBOSOME	SORT
	UPCASE	VERSION			
PIPLANE	FATAL			
PISCF	JACOBI	OVERLAP	PIALTER	PITILT
PITILT	OVERLAP	PIMOVE		
PLACE	DIST2	ERROR	GETPRB	GETTOR	INEDGE
POTNRG	GRADIENT			
POWER	RANDOM			
PRECOND	CHOLESKY	COLUMN		
PRESSURE	LATTICE			
PRMKEY	GETTEXT	GETWORD	POTOFF	UPCASE
PROJCT	DOT			
PROPERTY	IMPOSE			
PROTEIN	BASEFILE	CONNECT	DELETE	FIELD	FINAL
	FREEUNIT	GETKEY	GETSEQ	INITIAL	MAKEINT
	MAKEXYZ	NEXTARG	PROTEUS	PRTINT	PRTSEQ
	PRTXYZ	TRIMTEXT	VERSION		
PROTEUS	GETTEXT	SIDECHAIN	UPCASE	ZATOM
PRTBIOS	VERSION			
PRTDYN	FREEUNIT			
PRTERR	FREEUNIT	PRTINT	PRTXYZ	VERSION
PRTINT	VERSION			
PRTMOL2	VERSION			

PRTPDB	VERSION				
PRTPRM	NUMBER				
PRTSEQ	VERSION				
PRTXMOL	VERSION				
PRTXYZ	VERSION				
PSS	ACTIVE	FINAL	GETTEXT	GETXYZ	IMPOSE
	INITIAL	INITROT	MAKEINT	MAKEREF	MECHANIC
	MODECART	MODETORS	NEXTARG	PSSMIN	PSSWRITE
	SIGMOID	UPCASE			
PSS1	GRADIENT				
PSS2	HESSIAN				
PSSMIN	PSS1	PSS2	TNCG	WRITEOUT	
PSSRGD1	GRADRGD	RIGIDXYZ			
PSSRIGID	FINAL	FREEUNIT	GETTEXT	GETXYZ	IMPOSE
	INITIAL	MAKEREF	MECHANIC	NEXTARG	NUMERAL
	OCVM	ORIENT	PRTXYZ	PSSRGD1	RGDSRCH
	RIGIDXYZ	SIGMOID	UPCASE	VERSION	WRITEOUT
PSSROT	FINAL	GETINT	GETTEXT	IMPOSE	INITIAL
	INITROT	MAKEREF	MAKEXYZ	MECHANIC	MODETORS
	NEXTARG	OCVM	PSSROT1	PSSWRITE	UPCASE
	WRITEOUT				
PSSROT1	GRADROT	MAKEXYZ			
PSSWRITE	FREEUNIT	NUMERAL	PRTXYZ	VERSION	
PSSWRITE	FREEUNIT	NUMERAL	PRTXYZ	VERSION	
PTINCY	DOT	EPUCLC	PROJCT	ROTANG	
QUATFIT	JACOBI				
RANDOM	CALENDAR	GETTEXT	UPCASE		
RANVEC	RANDOM				
RATTLE	FATAL	IMAGE	PRTERR		
RATTLE2	FATAL	IMAGE	PRTERR		
READBLK	FREEUNIT	NUMERAL			

READDYN	FATAL	VERSION			
READINT	FATAL VERSION	GETTEXT	GETWORD	NEXTTEXT	TRIMTEXT
READMOL2	FATAL UPCASE	GETTEXT VERSION	GETWORD	SORT	TRIMTEXT
READPDB	FATAL UPCASE	FIXPDB VERSION	GETTEXT	GETWORD	TRIMTEXT
READPRM	FATAL NUMERAL	GETNUMB PRMKEY	GETSTRING TORPHASE	GETTEXT TRIMTEXT	GETWORD UPCASE
READSEQ	FATAL	GETNUMB	GETTEXT	TRIMTEXT	VERSION
READXYZ	FATAL TRIMTEXT	GETTEXT VERSION	GETWORD	NEXTTEXT	SORT
REFINE	INITER	LMQN	MIDERR	TOTERR	WRITEOUT
RESTRAIN	BNDLENG	DIHEDRAL	FATAL	GETTEXT	UPCASE
RGDSRCH	CLIMBRGD	EIGENRGD	RIGIDXYZ		
RIBOSOME	ADDBOND NEWATM	ADDSIDE OLDATM	FATAL PRTSEQ	FINDATM VERSION	FREEUNIT
RINGS	ANGLES	BONDS	FATAL	TORSIONS	
RMSERROR	TRIMTEXT				
ROTANG	DOT	VCROSS			
ROTCHECK	ROTLIST				
ROTLIST	FATAL				
SADDLE	FATAL IMPOSE NEXTARG READXYZ VERSION	FINAL INITIAL PATHPNT SADDLE1	FREEUNIT MAKEINT PATHSCAN SEARCH	GETTEXT MAKEXYZ PATHVAL TANGENT	GETXYZ MECHANIC PRTXYZ UPCASE
SADDLE1	GRADIENT				
SADDLES	ERROR	IPEDGE	TRIPLE		
SCAN	ACTIVE INITROT MODESRCH	FINAL LOCALMIN NEXTARG	FREEUNIT MAKEINT NUMERAL	GETXYZ MAPCHECK READXYZ	INITIAL MECHANIC VERSION

SCAN1	GRADIENT			
SCAN2	HESSIAN			
SDAREA	SURFATOM			
SDSTEP	GRADIENT	MDSTAT	RATTLE	RATTLE2	SDAREA
	SDTERM	TEMPER			
SDTERM	NORMAL			
SEARCH	FGVALUE			
SETIME	CLOCK			
SHAKEUP	GETNUMB	GETTEXT	GETWORD	UPCASE
SIDECHAIN	ZATOM			
SLATER	ASET	BSET	CJKM	POLYP
SMOOTH	GETTEXT	NEXTARG	UPCASE	
SNIFFER	FINAL	FREEUNIT	GETREF	GETXYZ	GRADIENT
	INITIAL	MAKEREF	MECHANIC	NEXTARG	PRTXYZ
	SNIFFER1	VERSION	WRITEOUT		
SNIFFER1	GRADIENT			
SOAK	DELETE	FREEUNIT	IMAGE	LATTICE	MAKEREF
	MERGE	MOLECULE	READXYZ	SUFFIX	UNITCELL
	VERSION				
SOLVATE	GETTEXT	UPCASE		
SPACEFILL	ACTIVE	CONNOLLY	FIELD	FINAL	GETTEXT
	GETXYZ	INITIAL	KATOM	KVDW	NEXTARG
	UPCASE				
SQUARE	GETTEXT	LMSTEP	PRECISE	QRFACT	RSDVALUE
	TRUST	UPCASE	WRITEOUT		
SUFFIX	TRIMTEXT			
SUPERPOSE	FIELD	FINAL	FREEUNIT	GETTEXT	GETXYZ
	IMPOSE	INITIAL	KATOM	PRTXYZ	UPCASE
	VERSION				
SURFACE	FATAL	SORT2		
SURFATOM	FATAL	SORT2		

SYBYLXYZ	FINAL VERSION	FREEUNIT	GETMOL2	INITIAL	PRTXYZ
SYMMETRY	CELLATOM				
TANGENT	PATHPNT	SADDLE1			
TESTGRAD	ENERGY INITIAL	FINAL MECHANIC	GETTEXT NEXTARG	GETXYZ UPCASE	GRADIENT
TESTHESS	ENERGY GRADIENT NUMGRAD	FINAL HESSIAN UPCASE	FREEUNIT INITIAL VERSION	GETTEXT MECHANIC	GETXYZ NEXTARG
TESTLIGHT	EBUCK ECHARGE1 EHAL4 ELJ5 FINAL MECHANIC	EBUCK1 ECHARGE4 EHAL5 EMM3HB GETIME NEXTARG	EBUCK4 ECHARGE5 ELJ EMM3HB1 GETXYZ SETIME	EBUCK5 EHAL ELJ1 EMM3HB4 INITIAL	ECHARGE EHAL1 ELJ4 EMM3HB5 LIGHTS
TESTROT	ENERGY INITROT	FINAL MAKEXYZ	GETINT MECHANIC	GRADROT NEXTARG	INITIAL
TIMER	ENERGY GRADIENT SETIME	FINAL HESSIAN UPCASE	GETIME INITIAL	GETTEXT MECHANIC	GETXYZ NEXTARG
TIMEROT	ENERGY GRADROT NEXTARG	FINAL HESSROT SETIME	GETIME INITIAL UPCASE	GETINT INITROT	GETTEXT MECHANIC
TNCG	BORN SEARCH	FGVALUE TNSOLVE	GETTEXT UPCASE	HMATRIX WRITEOUT	PISCF
TNSOLVE	FGVALUE	PRECOND			
TORSIONS	FATAL				
TORUS	ERROR	GETTOR			
TOTERR	BNDERR	CHIRER	LOCERR	TORSER	VDWERR
TRIANGLE	FATAL				
TRIPLE	DOT	VCROSS			
TRUST	PRECISE	RSDVALUE			
UNITCELL	GETTEXT	GETWORD	UPCASE		

VAM	CIRPLN	DEPTH	DIST2	DOT	ERROR
	GENDOT	MEASFN	MEASFP	MEASFS	MEASPM
	TRIPLE	VCROSS	VNORM		
VDWERR	LIGHTS				
VECANG	ANORM	DOT	TRIPLE		
VERLET	GRADIENT	MDSTAT	PRESSURE	RATTLE	RATTLE2
	TEMPER				
VERSION	LOWCASE	NEXTARG	TRIMTEXT		
VIBRATE	DIAGQ	FATAL	FINAL	FREEUNIT	GETXYZ
	HESSIAN	INITIAL	MECHANIC	NEXTARG	NUMERAL
	PRTXYZ	VERSION			
VIBROT	DIAGQ	FINAL	GETINT	HESSROT	INITIAL
	INITROT	MECHANIC			
VNORM	ANORM				
VOLUME	CONNOLLY				
VOLUME1	FATAL				
VOLUME2	FATAL				
WRITEOUT	FREEUNIT	NUMERAL	PRTINT	PRTXYZ	VERSION
XTALERR	ENERGY	XTALMOVE	XTALPRM		
XTALFIT	FINAL	GETXYZ	INITIAL	MECHANIC	NEXTARG
	SQUARE	XTALERR	XTALPRM	XTALWRT	
XTALLAT1	ENERGY	LATTICE			
XTALMIN	FINAL	FREEUNIT	GETXYZ	GRADIENT	INITIAL
	LATTICE	MECHANIC	NEXTARG	OCVM	PRTXYZ
	TNCG	VERSION	WRITEOUT	XTALLAT1	XTALMOL1
	XTALMOL2				
XTALMOL1	GRADIENT				
XTALMOL2	HESSIAN				
XTALMOVE	LATTICE				
XTALPRM	BOUNDS	LATTICE	MOLECULE		
XYZEDIT	ACTIVE	CUTOFFS	DELETE	FIELD	FINAL
	FREEUNIT	GETXYZ	INERTIA	INITIAL	INSERT

	KATOM	MAKEREF	MERGE	PRTXYZ	RANDOM
	SOAK	SORT	SORT4	VERSION	
XYZINT	FINAL	FREEUNIT	GETTEXT	GETXYZ	INITIAL
	MAKEINT	NEXTARG	PRTINT	READINT	UPCASE
	VERSION				
XYZPDB	FIELD	FINAL	FREEUNIT	GETXYZ	INITIAL
	KATOM	MAKEPDB	MOLECULE	PRTPDB	VERSION
XYZRIGID	JACOBI	ROTEULER			
XYZSYBYL	BONDS	FINAL	FREEUNIT	GETXYZ	INITIAL
	PRTMOL2	VERSION			
ZATOM	FATAL				
ZVALUE	MAKEXYZ	TRIMTEXT			

12. Examples using the TINKER Package

This section contains some comments on the five sample calculations found in the EXAMPLE subdirectory of the TINKER distribution. These examples exercise many of the current TINKER programs and provide a flavor of the capabilities of the present package.

ANION Example

Computes an estimation of the free energy of hydration of Cl^- anion vs. Br^- anion via a 1 picosecond simulation on a "hybrid" anion in a box of water, followed by free energy perturbation

ARGON Example

Performs a minimization followed by 6 picoseconds of a molecular dynamics run on a periodic box containing 150 argon atoms

CLUSTER Example

Performs a set of 10 Gaussian density annealing trials on a cluster of 13 argon atoms in an attempt to locate the "global" minimum energy structure

CRAMBIN Example

Generates a TINKER file from a PDB file, followed by a single point energy computation and determination of the molecular volume and surface area

CYCLOHEX Example

Locates the transition state between chair and boat cyclohexane, with subsequent refinement of the transition state and vibrational analysis to show one negative frequency

ENKEPHALIN Example

Produces coordinates from the amino acid sequence and phi/psi angles, followed by energy minimization and determination of the lowest frequency normal mode

FORMAMIDE Example

Converts to a unit cell from fractional coordinates, followed by full crystal energy minimization and determination of optimal carbonyl oxygen energy parameters from a fit to lattice energy and structure

HELIX Example

Performs a rigid-body optimization of the packing of two idealized polyalanine helices using only van der Waals interactions

13. Benchmark Results

The tables in this section provide CPU benchmarks for basic TINKER energy evaluations. All times are in seconds and were measured with TINKER 3.7, dimensioned to MAXATM of 10000 and MAXHESS of 1000000. Each benchmark was run twice in rapid succession on a quiet machine. Times reported are the fastest results from the second run. Please note that many of the machines listed are based on older generation CPUs. If you are able to run the benchmarks on additional machine types, please send the results for inclusion in a future listing.

Isolated Crambin Molecule – Potential Energy Evaluation

The system is a single Crambin protein molecule (46 amino acids, 642 atoms) evaluated at the X-ray crystal conformation using the TINKER MM3PRO force field, a protein only version of MM3, with a 9 Å cutoff for vdw, 12 Å for dipole and 20 Å for charge interactions.

MACHINE TYPE	MHz	SETUP	ENERGY	GRAD	HESSIAN
Aspen Durango II (Alpha/Linux)	533	0.120	0.086	0.134	1.077
DEC Alpha 4100 (Tru64 4.0E)	400	0.152	0.114	0.173	1.309
DEC Alpha 200 4/233 (Tru64 4.0E)	233	0.451	0.303	0.586	4.823
SGI Challenge R10k (Irix 6.2)	195	0.314	0.130	0.210	1.737
SGI IndigoII R4k (Irix 6.2)	200	0.744	0.407	0.751	7.577
Sun Ultra-1 Enterprise (SunOS 5.5)	167	0.378	0.237	0.411	3.275
Dell Dimension XPS R450 (W98)	450	0.270	0.110	0.160	1.430
Digital HiNote Ultra 2000 (W95)	233	0.710	0.380	0.610	5.770
Digital HiNote Ultra 2000 (Linux)	233	0.690	0.290	0.560	5.880
Power Macintosh G3 (MacOS 8.5)	300	0.283	0.200	0.300	2.800
PowerCenter Pro 604e (MacOS 8.5)	180	0.500	0.375	0.500	5.750

Crambin Crystal – Potential Energy Evaluation

The system is the X-ray unit cell of Crambin under periodic boundary conditions (two protein molecules, two ethanol and 178 TIP3P water; total of 1334 atoms) evaluated using the OPLS-UA force field with a 9 Å cutoff on vdw interactions and PM Ewald for charge interactions.

MACHINE TYPE	MHz	SETUP	ENERGY	GRAD	HESSIAN
Aspen Durango II (Alpha/Linux)	533	0.212	0.630	0.802	2.733
DEC Alpha 4100 (Tru64 4.0E)	400	0.256	0.826	1.024	3.247
DEC Alpha 200 4/233 (Tru64 4.0E)	233	0.780	2.194	3.034	10.419
SGI Challenge R10k (Irix 6.2)	195	0.525	0.946	1.301	3.606
SGI IndigoII R4k (Irix 6.2)	200	1.167	2.832	3.995	12.757
Sun Ultra-1 Enterprise (SunOS 5.5)	167	0.729	1.701	2.505	7.530
Dell Dimension XPS R450 (W98)	450	0.490	0.880	1.090	3.570
Digital HiNote Ultra 2000 (W95)	233	1.320	2.420	3.460	12.570
Digital HiNote Ultra 2000 (Linux)	233	1.160	2.630	3.710	12.040
Power Macintosh G3 (MacOS 8.5)	300	0.517	1.467	1.967	5.967
PowerCenter Pro 604e (MacOS 8.5)	180	0.875	2.375	3.250	12.250

216 TIP3P Waters in Periodic Box – 1000 Dynamics Steps

The system is a preequilibrated collection of 216 TIP3P water molecules in a periodic box with an edge length of 18.6206 Å and a cutoff of 9 Å on nonbonded interactions. The timed simulation is 1000 steps of standard MD at 300K with a time step of 1.0 femtosecond.

MACHINE TYPE	MHz	DYNAMICS
Aspen Durango II (Alpha/Linux)	533	113
DEC Alpha 4100 (Tru64 4.0E)	400	138
DEC Alpha 200 4/233 (Tru64 4.0E)	233	714
SGI Challenge R10k (Irix 6.2)	195	147
SGI IndigoII R4k (Irix 6.2)	200	488
Sun Ultra-1 Enterprise (SunOS 5.5)	167	279
Dell Dimension XPS R450 (W98)	450	126
Digital HiNote Ultra 2000 (W95)	233	672
Digital HiNote Ultra 2000 (Linux)	233	466
Power Macintosh G3 (MacOS 8.5)	300	225
PowerCenter Pro 604e (MacOS 8.5)	180	373

14. Collaborators & Acknowledgments

The TINKER package has developed over a period of many years, very slowly through the mid- to late-1980's, and more rapidly since the early 1990's in Jay Ponder's research group at the Washington University School of Medicine in St. Louis. Many people have played significant roles in the development of the package into its current form. The major contributors are listed below:

Stew Rubenstein   coordinate interconversions; original optimization methods and torsional angle manipulation

Craig Kundrot   molecular surface area & volume and their derivatives

Shawn Huston   original AMBER/OPLS implementation; free energy calculations; time correlation functions

Mike Dudek   DMA-derived multipole models for peptides and proteins

Yong "Mike" Kong   multipole electrostatics; dipole polarization; reaction field treatment; TINKER water model

Reece Hart   potential smoothing methodology; Scheraga's DEM, Straub's GDA and extensions

Mike Hodsdon   extension of the TINKER distgeom program and its application to NMR NOE structure determination

Rohit Pappu   potential smoothing methodology and PSS algorithms; rigid body optimization

Wijnand Mooij   MM3 directional hydrogen bonding term; crystal minimization code

Gerald Loeffler   stochastic/Langevin dynamics implementation

In addition, we would like to thank **Tom Darden** for making his particle mesh Ewald code generally available to the simulation community.

It is critically important that TINKER's distributed force field parameter sets exactly reproduce the intent of the original force field authors. We would like to thank **Julian Tirado-Rives** (OPLS-AA), **Alex MacKerell** (CHARMM22), and **Adrian Roitberg** and **Carlos Simmerling** (AMBER) for their help in testing TINKER's results against those given by the authentic programs and parameter sets.

Finally, we wish to thank the many users of the TINKER package for their suggestions and comments, praise and criticism, which have resulted in a variety of improvements.

15.   References & Suggested Reading

This section contains a list of the references to algorithms and procedures which have been of use during the development of TINKER. Methods described in some of the references have been implemented in detail within the TINKER source code. Other references contain useful background information although the algorithms themselves are now obsolete. Still other papers contain ideas or extensions planned for future inclusion in TINKER. The list is heavily skewed toward biomolecules in general and proteins in particular. This bias reflects our group's major interests; however an attempt has been made to include methods which should be generally applicable.

PARTIAL LIST OF MOLECULAR MECHANICS SOFTWARE PACKAGES

AMBER	Peter Kollman, University of California, San Francisco
AMMP	Rob Harrison, Thomas Jefferson University, Philadelphia
ARGOS	J. Andrew McCammon, Univ. of California, San Diego
BOSS	William Jorgensen, Yale University
BRUGEL	Shoshona Wodak, Free University of Brussels
CFF	Shneior Lifson, Weizmann Institute
CHARMM	Martin Karplus, Harvard University
CHARMM/GEMM	Bernard Brooks, National Institutes of Health, Bethesda
DELPHI	Bastian van de Graaf, Delft University of Technology
DISCOVER	Arne Hagler, BIOSYM Technologies/Molecular Simulations
DL POLY	W. Smith & T. Forester, CCP5, Daresbury Laboratory
ECEPP	Harold Scheraga, Cornell University
ENCAD	Michael Levitt, Stanford University
FANTOM	Werner Braun, University of Texas, Galveston
FEDER/2	Nobuhiro Go, Kyoto University
GROMACS	Herman Berendsen, University of Groningen
GROMOS	Wilfred van Gunsteren, BIOMOS and ETH, Zurich
IMPACT	Ronald Levy, Rutgers University
MACROMODEL	W. Clark Still, Columbia University
MM2/MM3/MM4	N. Lou Allinger, University of Georgia
MMC	Cliff Dykstra, Indiana Univ.–Purdue Univ. at Indianapolis
MMFF	Tom Halgren, Merck Research Laboratories, Rahway
MMTK	Konrad Hinsen, Inst. of Structural Biology, Grenoble
MOIL	Ron Elber, The Hebrew University, Jerusalem
MOLARIS	Arieh Warshal, University of Southern California
MOLDY	Keith Refson, Oxford University
MOSCITO	Dietmar Paschek & Alfons Geiger, Unversity of Dortmund
NAMD2	Klaus Schulten, University of Illinois, Urbana
OOMPAA	Andy McCammon, University of California, San Diego
ORAL	Karel Zimmerman, INRA, Jouy-en-Josas, France
ORIENT	Anthony Stone, Cambridge University
PEFF	Jan Dillen, University of Pretoria, South Africa
Q	Johan Åqvist, Uppsala University
SIBFA	Nohad Gresh, INSERM, CNRS, Paris
SIGMA	Jan Hermans, University of North Carolina
SPASIBA	Gerard Vergoten, Université de Lille
SPASMS	David Spellmeyer and the Kollman Group, UCSF
TINKER	Jay Ponder, Washington University, St. Louis

XPLOR/CNS   Axel Brunger, Yale University
YAMMP   Stephen Harvey, University of Alabama, Birmingham
YASP   Florian Mueller-Plathe, ETH Zentrum, Zurich
YETI   Angelo Vedani, University of Kansas

AMBER D. A Pearlman, D. A. Case, J. W. Caldwell, W. S. Ross, T. E. Cheatham III, S. DeBolt, D. Ferguson, G. Seibel and P. Kollman, AMBER, a Package of Computer Programs for Applying Molecular Mechanics, Normal Mode Analysis, Molecular Dynamics and Free Energy Calculations to Simulate the Structural and Energetic Properties of Molecules, **_Comp. Phys. Commun._**, _91_, 1-41 (1995)

ARGOS T. P. Straatsma and J. A. McCammon, ARGOS, a Vectorized General Molecular Dynamics Program, **_J. Comput. Chem._**, _11_, 943-951 (1990)

CHARMM B. R. Brooks, R. E. Bruccoleri, B. D. Olafson, D. J. States, S. Swaminathan and M. Karplus, CHARMM: A Program for Macromolecular Energy, Minimization, and Dynamics Calculations, **_J. Comput. Chem._**, _4_, 187-217 (1983)

ENCAD M. Levitt, M. Hirshberg, R. Sharon and V. Daggett, Potential Energy Function and Parameters for Simulations for the Molecular Dynamics of Proteins and Nucleic Acids in Solution, **_Comp. Phys. Commun._**, _91_, 215-231 (1995)

FANTOM T. Schaumann, W. Braun and K. Wurtrich, The Program FANTOM for Energy Refinement of Polypeptides and Proteins Using a Newton-Raphson Minimizer in Torsion Angle Space, **_Biopolymers_**, _29_, 679-694 (1990)

FEDER/2 H. Wako, S. Endo, K. Nagayama and N. Go, FEDER/2: Program for Static and Dynamic Conformational Energy Analysis of Macro-molecules in Dihedral Angle Space, **_Comp. Phys. Commun._**, _91_, 233-251 (1995)

GROMACS H. J. C. Berendsen, D. van der Spoel and R. van Drunen, GROMACS: A Message-passing Parallel Molecular Dynamics Implementation, **_Comp. Phys. Commun._**, _91_, 43-56 (1995)

IMPACT D. B. Kitchen, F. Hirata, J. D. Westbrook, R. Levy, D. Kofke and M. Yarmush, Conserving Energy during Molecular Dynamics Simulations of Water, Proteins, and Proteins in Water, **_J. Comput. Chem._**, _10_, 1169-1180 (1990)

MACROMODEL F. Mahamadi, N. G. J. Richards, W. C. Guida, R. Liskamp, M. Lipton, C. Caufield, G. Chang, T. Hendrickson and W. C. Still, MacroModel–An Integrated Software System for Modeling Organic and Bioorganic Molecules Using Molecular Mechanics, **_J. Comput. Chem._**, _11_, 440-467 (1990)

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)

MM3 N. L. Allinger, Y. H. Yuh and J.-H. Lii, Molecular Mechanics. The MM3 Force Field for Hydrocarbons, **_J. Am. Chem. Soc._**, _111_, 8551-8566 (1989)

MM4 N. L. Allinger, K. Chen and J.-H. Lii, An Improved Force Field (MM4) for Saturated Hydrocarbons, **_J. Comput. Chem._**, _17_, 642-668 (1996)

MMC   C. E. Dykstra, Molecular Mechanics for Weakly Interacting Assemblies of Rare Gas Atoms and Small Molecules, *J. Am. Chem. Soc.*, *111*, 6168-6174 (1989)

MMFF   T. A. Halgren, Merck Molecular Force Field. I. Basis, Form, Scope, Parameterization, and Performance of MMFF94, *J. Comput. Chem.*, *17*, 490-516 (1996)

MOIL   R. Elber, A. Roitberg, C. Simmerling, R. Goldstein, H. Li, G. Verkhiver, C. Keasar, J. Zhang and A. Ulitsky, MOIL: A Program for Simulations of Macromolecules, *Comp. Phys. Commun.*, *91*, 159-189 (1995)

NAMD2   L. Kalé, R. Skeel, M. Bhandarkar, R. Brunner, A. Gursoy, N. Krawetz, J. Phillips, A. Shinozaki, K. Varadarajan and K. Schulten, NAMD2: Greater Scalability for Parallel Molecular Dynamics, *J. Comput. Phys.*, *151*, 283-312 (1999)

OOMPAA   G. A. Huber and J. A. McCammon, OOMPAA–Object-oriented Model for Probing Assemblages of Atoms, *J. Comput. Phys.*, *151*, 264-282 (1999)

ORAL   K. Zimmermann, ORAL: All Purpose Molecular Mechanics Simulator and Energy Minimizer, *J. Comput. Chem.*, *12*, 310-319 (1991)

PEFF   J. L. M. Dillen, PEFF: A Program for the Development of Empirical Force Fields, *J. Comput. Chem.*, *13*, 257-267 (1992)

SIBFA   N. Gresh, Inter- and Intramolecular Interactions. Inception and Refinements of the SIBFA, Molecular Mechanics (SMM) Procedure, a Separable, Polarizable Methodology Grounded on *ab Initio* SCF/MP2 Computations. Examples of Applications to Molecular Recognition Problems, *J. Chim. Phys. PCB*, *94*, 1365-1416 (1997)

SPASIBA   P. Derreumaux and G. Vergoten, A New Spectroscopic Molecular Mechanics Force-Field - Parameters For Proteins, *J. Chem. Phys.*, *102*, 8586-8605 (1995)

YAMMP   R. K.-Z. Tan and S. C. Harvey, Yammp: Development of a Molecular Mechanics Program Using the Modular Programming Method, *J. Comput. Chem.*, *14*, 455-470 (1993)

YETI   A. Vedani, YETI: An Interactive Molecular Mechanics Program for Small-Molecule Protein Complexes, *J. Comput. Chem.*, *9*, 269-280 (1988)

MOLECULAR MECHANICS

U. Burkert and N. L. Allinger, **Molecular Mechanics**, American Chemical Society, Washington, D.C., 1982.

K. Rasmussen, **Potential Energy Functions in Conformational Analysis** (Lecture Notes in Chemistry, Vol. 27), Springer-Verlag, Berlin, 1985.

A. K. Rappé and C. J. Casewit, **Molecular Mechanics across Chemistry**, University Science Books, Sausalito, CA, 1997.

P. Comba and T. W. Hambley, **Molecular Modeling of Inorganic Compounds**, VCH, New York, 1995.

COMPUTER SIMULATION METHODS

M. P. Allen and D. J. Tildesley, **Computer Simulation of Liquids**, Oxford University Press, Oxford, 1987.

A. R. Leach, **Molecular Modelling: Principles and Applications**, Addison Wesley Longman, Essex, England, 1996.

D. Frankel and B. Smit, **Understanding Molecular Simulation: From Algorithms to Applications**, Academic Press, San Diego, CA, 1996.

D. C. Rapaport, **The Art of Molecular Dynamics Simulation**, Cambridge University Press, Cambridge, 1995.

J. M. Haile, **Molecular Dynamics Simulation**, John Wiley and Sons, New York, 1992.

D. W. Heermann, **Computer Simulation Methods in Theoretical Physics**, Springer-Verlag, Berlin, 1986.

T. Schlick, R. D. Skeel, A. T. Brünger, L. V. Kale, J. A. Board, J. Hermans and K. Schulten, Algorithmic Challenges in Computational Molecular Biophysics, **_J. Comput. Phys._**, _151_, 9-48 (1999)

MODELING OF BIOLOGICAL MACROMOLECULES

J. A. McCammon and S. Harvey, **Dynamics of Proteins and Nucleic Acids**, Cambridge University Press, Cambridge, 1987.

C. L. Brooks III, M. Karplus and B. M. Pettitt, **Proteins: A Theoretical Perspective of Dynamics, Structure, and Thermodynamics**, John Wiley and Sons, New York, 1988.

W. F. van Gunsteren, P. K. Weiner and A. J. Wilkinson, **Computer Simulation of Biomolecular Systems, Vol. 1-3**, Kluwer Academic Publishers, Dordrecht, 1989-1997.

T. E. Cheatham and B. R. Brooks, Recent advances in molecular dynamics simulation towards the realistic representation of biomolecules in solution, **_Theor. Chem. Acc._**, _99_, 279-288 (1998)

TRUNCATED NEWTON OPTIMIZATION

J. W. Ponder and F. M. Richards, An Efficient Newton-like Method for Molecular Mechanics Energy Minimization of Large Molecules, **_J. Comput. Chem._**, _8_, 1016-1024 (1987)

R. S. Dembo and T. Steihaug, Truncated-Newton Algorithms for Large-Scale Unconstrained Optimization, **_Math. Prog._**, _26_, 190-212 (1983)

S. C. Eisenstat and H. F. Walker, Choosing the Forcing Terms in an Inexact Newton Method, **_SIAM J. Sci. Comput._**, _17_, 16-32 (1996)

T. Schlick and M. Overton, A Powerful Truncated Newton Method for Potential Energy Minimization, **_J. Comput. Chem._**, _8_, 1025-1039 (1987)

D. S. Kershaw, The Incomplete Cholesky-Conjugate Gradient Method for the Iterative Solution of Systems of Linear Equations, *J. Comput. Phys.*, *26*, 43-65 (1978)

T. A. Manteuffel, An Incomplete Factorization Technique for Positive Definite Linear Systems, *Math. Comp.*, *34*, 473-497 (1980)

P. Derreumaux, G. Zhang and T. Schlick and B. R. Brooks, A Truncated Newton Minimizer Adapted for CHARMM and Biomolecular Applications, *J. Comput. Chem.*, *15*, 532-552 (1994)

I. S. Duff, A. M. Erisman and J. K. Reid, **Direct Methods for Sparse Matrices**, Oxford University Press, Oxford, 1986.

NONLINEAR CONJUGATE GRADIENT OPTIMIZATION

D. G. Luenberger, **Linear and Nonlinear Programming**, 2nd Ed., Addison-Wesley, Reading, MA, 1984.

S. J. Watowich, E. S. Meyer, R. Hagstrom and R. Josephs, A Stable, Rapidly Converging Conjugate Gradient Method for Energy Minimization, *J. Comput. Chem.*, *9*, 650-661 (1988)

QUASI-NEWTON OPTIMIZATION

P. E. Gill, W. Murray and M. H. Wright, **Practical Optimization**, Academic Press, New York, 1981.

W. C. Davidon, Optimally Conditioned Optimization Algorithms without Line Searches, *Math. Prog.*, *9*, 1-30 (1975)

J. Nocedal, Updating Quasi-Newton Matrices with Limited Storage, *Math. Comp.*, 773-782 (1980)

"SNIFFER" GLOBAL OPTIMIZATION

A. O. Griewank, Generalized Descent for Global Optimization, *J. Opt. Theor. Appl.*, *34*, 11-39 (1981)

R. A. R. Butler and E. E. Slaminka, An Evaluation of the Sniffer Global Optimization Algorithm Using Standard Test Functions, *J. Comput. Phys.*, *99*, 28-32 (1993)

J. W. Rogers and R. A. Donnelly, Potential Transformation Methods for Large-Scale Global Optimization, *SIAM J. Optim.*, *5*, 871-891 (1995)

POTENTIAL ENERGY SMOOTHING

R. V. Pappu, R. K. Hart and J. W. Ponder, Analysis and Application of Potential Energy Smoothing Methods for Global Optimization, *J. Phys. Chem. B*, *102*, 9725-9742 (1998)

L. Piela, J. Kostrowicki and H. A. Scheraga, The Multiple-Minima Problem in the Conformational Analysis of Molecules. Deformation of the Potential Energy Hypersurface by the Diffusion Equation Method, *J. Phys. Chem.*, *93*, 3339-3346 (1989)

J. Ma and J. E. Straub, Simulated Annealing Using the Classical Density Distribution, ***J. Chem. Phys.***, *101*, 533-541 (1994)

C. Tsoo and C. L. Brooks, Cluster Structure Determination Using Gaussian Density Distribution Global Minimization Methods, ***J. Chem. Phys.***, *101*, 6405-6411 (1994)

S. Nakamura, H. Hirose, M. Ikeguchi and J. Doi, Conformational Energy Minimization Using a Two-Stage Method, ***J. Phys. Chem.***, *99*, 8374-8378 (1995)

T. Huber, A. E. Torda and W. F. van Gunsteren, Structure Optimization Combining Soft-Core Interaction Functions, the Diffusion Equation Method, and Molecular Dynamics, ***J. Phys. Chem. A***, *101*, 5926-5930 (1997)

S. Schelstraete and H. Verschelde, Finding Minimum-Energy Configurations of Lennard-Jones Clusters Using an Effective Potential, ***J. Phys. Chem. A***, *101*, 310-315 (1998)

I. Andricioaei and J. E. Straub, Global Optimization Using Bad Derivatives: Derivative-Free Method for Molecular Energy Minimization, ***J. Comput. Chem.***, *19*, 1445-1455 (1998)

L. Piela, Search for the Most Stable Structures on Potential Energy Surfaces, ***Coll. Czech. Chem. Commun.***, *63*, 1368-1380 (1998)

INTEGRATION METHODS FOR MOLECULAR DYNAMICS

D. Beeman, Some Multistep Methods for Use in Molecular Dynamics Calculations, ***J. Comput. Phys.***, *20*, 130-139 (1976)

M. Levitt and H. Meirovitch, Integrating the Equations of Motion, ***J. Mol. Biol.***, *168*, 617-620 (1983)

J. Aqvist, W. F. van Gunsteren, M. Leijonmarck and O. Tapia, A Molecular Dynamics Study of the C-Terminal Fragment of the L7/L12 Ribosomal Protein, ***J. Mol. Biol.***, *183*, 461-477 (1985)

W. C. Swope, H. C. Andersen, P. H. Berens and K. R. Wilson, A Computer Simulation Method for the Calculation of Equilibrium Constants for the Formation of Physical Clusters of Molecules: Application to Small Water Clusters, ***J. Chem. Phys.***, *76*, 637-649 (1982)

LANGEVIN, BROWNIAN AND STOCHASTIC DYNAMICS

M. P. Allen, Brownian Dynamics Simulation of a Chemical Reaction in Solution, ***Mol. Phys.***, *40*, 1073-1087 (1980)

W. F. van Gunsteren and H. J. C. Berendsen, Algorithms for Brownian Dynamics, ***Mol. Phys.***, *45*, 637-647 (1982)

F. Guarnieri and W. C. Still, A Rapidly Convergent Simulation Method: Mixed Monte Carlo/Stochastic Dynamics, ***J. Comput. Chem.***, *15*, 1302-1310 (1994)

CONSTANT TEMPERATURE AND PRESSURE DYNAMICS

H. J. C. Berendsen, J. P. M. Postma, W. F. van Gunsteren, A. DiNola and J. R. Haak, Molecular Dynamics with Coupling to an External Bath, *J. Chem. Phys.*, *81*, 3684-3690 (1984)

W. G. Hoover, Canonical Dynamics: Equilibrium Phase-space Distributions, *Phys. Rev. A*, *31*, 1695-1697 (1985)

J. J. Morales, S. Toxvaerd and L. F. Rull, Computer Simulation of a Phase Transition at Constant Temperature and Pressure, *Phys. Rev. A*, *34*, 1495-1498 (1986)

B. R. Brooks, Algorithms for Molecular Dynamics at Constant Temperature and Pressure, Internal Report of Division of Computer Research and Technology, National Institutes of Health, 1988.

M. L. Levitt, Molecular Dynamics of Native Protein: Computer Simulation of Trajectories, *J. Mol. Biol.*, *168*, 595-620 (1983)

ANALYTICAL DERIVATIVES OF POTENTIAL FUNCTIONS

K. J. Miller, R. J. Hinde and J. Anderson, First and Second Derivative Matrix Elements for the Stretching, Bending, and Torsional Energy, *J. Comput. Chem.*, *10*, 63-76 (1989)

D. H. Faber and C. Altona, UTAH5: A Versatile Programme Package for the Calculation of Molecular Properties by Force Field Methods, *Computers & Chemistry*, *1*, 203-213 (1977)

W. C. Swope and D. M. Ferguson, Alternative Expressions for Energies and Forces Due to Angle Bending and Torsional Energy, Report G320-3561, *J. Comput. Chem.*, *13*, 585-594 (1992)

A. Blondel and M. Karplus, New Formulation for Derivatives of Torsion Angles and Improper Torsion Angles in Molecular Mechanics: Elimination of Singularities, *J. Comput. Chem.*, *17*, 1132-1141 (1996)

R. E. Tuzun, D. W. Noid and B. G. Sumpter, Efficient Treatment of Out-of-Plane Bend and Improper Torsion Interactions in MM2, MM3, and MM4 Molecular Mechanics Calculations, *J. Comput. Chem.*, *18*, 1804-1811 (1997)

TORSIONAL SPACE DERIVATIVES AND NORMAL MODES

M. Levitt, C. Sander and P. S. Stern, Protein Normal-mode Dynamics: Trypsin Inhibitor, Crambin, Ribonuclease and Lysozyme, *J. Mol. Biol.*, *181*, 423-447 (1985)

M. Levitt, Protein Folding by Restrained Energy Minimization and Molecular Dynamics, *J. Mol. Biol.*, *170*, 723-764 (1983)

H. Wako and N. Go, Algorithm for Rapid Calculation of Hessian of Conformational Energy Function of Proteins by Supercomputer, *J. Comput. Chem.*, *8*, 625-635 (1987)

H. Abe, W. Braun, T. Noguti and N. Go, Rapid Calculation of First and Second Derivatives of Conformational Energy with Respect to Dihedral Angles for Proteins: General Recurrent Equations, *Computers & Chemistry*, *8*, 239-247 (1984)

T. Noguti and N. Go, A Method of Rapid Calculation of a Second Derivative Matrix of Conformational Energy for Large Molecules, *J. Phys. Soc. Japan*, *52*, 3685-3690 (1983)

MOLECULAR SURFACE AREA AND VOLUME

M. L. Connolly, Analytical Molecular Surface Calculation, *J. Appl. Cryst.*, *16*, 548-558 (1983)

M. L. Connolly, Computation of Molecular Volume, *J. Am. Chem. Soc.*, *107*, 1118-1124 (1985)

C. E. Kundrot, J. W. Ponder and F. M. Richards, Algorithms for Calculating Excluded Volume and Its Derivatives as a Function of Molecular Conformation and Their Use in Energy Minimization, *J. Comput. Chem.*, *12*, 402-409 (1991)

T. J. Richmond, Solvent Accessible Surface Area and Excluded Volume in Proteins, *J. Mol. Biol.*, *178*, 63-89 (1984)

L. Wesson and D. Eisenberg, Atomic Solvation Parameters Applied to Molecular Dynamics of Proteins in Solution, *Protein Science*, *1*, 227-235 (1992)

W. Hasel, T. F. Hendrickson and W. C. Still, A Rapid Approximation to the Solvent Accessible Surface Areas of Atoms, *Tetrahedron Comput. Method.*, *1*, 103-116 (1988)

K. D. Gibson and H. A. Scheraga, Exact Calculation of the Volume and Surface Area of Fused Hard-sphere Molecules with Unequal Atomic Radii, *Mol. Phys.*, *62*, 1247-1265 (1987)

K. D. Gibson and H. A. Scheraga, Surface Area of the Intersection of Three Spheres with Unequal Radii: A Simplified Analytical Formula, *Mol. Phys.*, *64*, 641-644 (1988)

S. Sridharan, A. Nichols and K. A. Sharp, A Rapid Method for Calculating Derivatives of Solvent Accessible Surface Areas of Molecules, *J. Comput, Chem.*, *16*, 1038-1044 (1995)

BOUNDARY CONDITIONS AND NEIGHBOR METHODS

W. F. van Gunsteren, H. J. C. Berendsen, F. Colonna, D. Perahia, J. P. Hollenberg and D. Lellouch, On Searching Neighbors in Computer Simulations of Macromolecular Systems, *J. Comput. Chem.*, *5*, 272-279 (1984)

F. Sullivan, R. D. Mountain and J. O'Connell, Molecular Dynamics on Vector Computers, *J. Comput. Phys.*, *61*, 138-153 (1985)

J. Boris, A Vectorized "Near Neighbors" Algorithm of Order N Using a Monotonic Logical Grid, *J. Comput. Phys.*, *66*, 1-20 (1986)

S. G. Lambrakos and J. P. Boris, Geometric Properties of the Monotonic Lagrangian Grid Algorithm for Near Neighbors Calculations, *J. Comput. Phys.*, *73*, 183-202 (1987)

T. A. Andrea, W. C. Swope and H. C. Andersen, The Role of Long Ranged Forces in Determining the Structure and Properties of Liquid Water, *J. Chem. Phys.*, *79*, 4576-4584 (1983)

D. N. Theodorou and U. W. Suter, Geometrical Considerations in Model Systems with Periodic Boundary Conditions, *J. Chem. Phys.*, *82*, 955-966 (1985)

J. Barnes and P. Hut, A Hierarchical O(NlogN) Force-calculation Algorithm, *Nature*, *234*, 446-449 (1986)

CUTOFF AND TRUNCATION METHODS

P. J. Steinbach and B. R. Brooks, New Spherical-Cutoff Methods for Long-Range Forces in Macromolecular Simulation, *J. Comput. Chem.*, *15*, 667-683 (1993)

R. J. Loncharich and B. R. Brooks, The Effects of Truncating Long-Range Forces on Protein Dynamics, *Proteins*, *6*, 32-45 (1989)

C. L. Brooks III, B. M. Pettitt and M. Karplus, Structural and Energetic Effects of Truncating Long Ranged Interactions in Ionic and Polar Fluids, *J. Chem. Phys.*, *83*, 5897-5908 (1985)

EWALD SUMMATION

T. Darden, D. York and L. G. Pedersen, Particle Mesh Ewald: An N·log(N) Method for Ewald Sums in Large Systems, *J. Chem. Phys.*, *98*, 10089-10092 (1993)

U. Essmann, L. Perera, M. L. Berkowitz, T. Darden, H. Lee and L. G. Pedersen, A Smooth Particle Mesh Ewald Method, *J. Chem. Phys.*, *103*, 8577-8593 (1995)

A. Y. Toukmaji and J. A. Board, Jr., Ewald Summation Techniques in Perspective: A Survey, *Comp. Phys. Commun.*, *95*, 73-92 (1996)

S. E. Feller, R. W. Pastor, A. Rojnuckarin, S. Bogusz and B. R. Brooks, Effect of Electrostatic Force Truncation on Interfacial and Transport Properties of Water, *J. Phys. Chem.*, *100*, 17011-17020 (1996)

T. Darden, L. Perera, L. Li and L. Pedersen, New Tricks for Modelers from the Crystallography Toolkit: The Particle Mesh Ewald Algorithm and its Use in Nucleic Acid Simulations, *Structure*, *7*, R550-R60 (1999)

CONJUGATED AND AROMATIC SYSTEMS

J. T. Sprague, J. C. Tai, Y. Yuh and N. L. Allinger, The MMP2 Calculational Method, *J. Comput. Chem.*, *8*, 581-603 (1987)

J. Kao, A Molecular Orbital Based Molecular Mechanics Approach to Study Conjugated Hydrocarbons, *J. Am. Chem. Soc.*, *109*, 3818-3829 (1987)

J. Kao and N. L. Allinger, Conformational Analysis: Heats of Formation of Conjugated Hydrocarbons by the Force Field Method, *J. Am. Chem. Soc.*, *99*, 975-986 (1977)

D. H. Lo and M. A. Whitehead, Accurate Heats of Atomization and Accurate Bond Lengths: Benzenoid Hydrocarbons, *Can. J. Chem.*, *46*, 2027-2040 (1968)

G. D. Zeiss and M. A. Whitehead, Hetero-atomic Molecules: Semi-empirical Molecular Orbital Calculations and Prediction of Physical Properties, *J. Chem. Soc. A*, 1727-1738 (1971)

FREE ENERGY SIMULATION METHODS

P. Kollman, Free Energy Calculations: Applications to Chemical and Biochemical Phenomena, **Chem. Rev.**, *93*, 2395-2417 (1993)

B. L. Tembe and J. A. McCammon, Ligand-Receptor Interactions, **Computers & Chemistry**, *8*, 281-283 (1984)

W. L. Jorgensen and C. Ravimohan, Monte Carlo Simulation of Differences in Free Energy of Hydration, **J. Chem. Phys.**, *83*, 3050-3054 (1985)

W. L. Jorgensen, J. K. Buckner, S. Boudon and J. Tirado-Rives, Efficient Computation of Absolute Free Energies of Binding by Computer Simulations: Application to the Methane Dimer in Water, **J. Chem. Phys.**, *89*, 3742-3746 (1988)

S. H. Fleischman and C. L. Brooks III, Thermodynamics of Aqueous Solvation: Solution Properties of Alcohols and Alkanes, **J. Chem. Phys.**, *87*, 3029-3037 (1987)

U. C. Singh, F. K. Brown, P. A. Bash and P. A. Kollman, An Approach to the Application of Free Energy Perturbation Methods Using Molecular Dynamics, **J. Am. Chem. Soc.**, *109*, 1607-1614 (1987)

D. A. Pearlman and P. A. Kollman, A New Method for Carrying out Free Energy Perturbation Calculations: Dynamically Modified Windows, **J. Chem. Phys.**, *90*, 2460-2470 (1989)

T. P. Straatsma, H. J. C. Berendsen and J. P. M. Postma, Free Energy of Hydrophobic Hydration: A Molecular Dynamics Study of Noble Gases in Water, **J. Chem. Phys.**, *85*, 6720-6727 (1986)

T.P. Straatsma and H.J.C. Berendsen, Free Energy of Ionic Hydration: Analysis of a Thermodynamic Integration Technique to Evaluate Free Energy Differences by Molecular Dynamics Simulations, **J. Chem. Phys.**, *89*, 5876-5886 (1988)

M. Mezei, The Finite Difference Thermodynamic Integration, Tested on Calculating the Hydration Free Energy Difference between Acetone and Dimethylamine in Water, **J. Chem. Phys.**, *86*, 7084-7088 (1987)

A. E. Mark and W. F. van Gunsteren, Decomposition of the Free Energy of a System in Terms of Specific Interactions, **J. Mol. Biol.**, *240*, 167-176 (1994)

S. Boresch and M. Karplus, The Meaning of Copmponent Analysis: Decomposition of the Free Energy in Terms of Specific Interactions, **J. Mol. Biol.**, *254*, 801-807 (1995)

MACROSCOPIC TREATMENT OF SOLVENT EFFECTS

D. Eisenberg and A. D. McLachlan, Solvation Energy in Protein Folding and Binding, **Nature**, *319*, 199-203 (1986)

T. Ooi, M. Oobatake, G. Nemethy and H. A. Scheraga, Accessible Surface Areas as a Measure of the Thermodynamic Parameters of Hydration of Peptides, **Proc. Natl. Acad. Sci. USA**, *84*, 3086-3090 (1987)

W. C. Still, A. Tempczyk, R. C. Hawley and T. Hendrickson, A Semiempirical Treatment of Solvation for Molecular Mechanics and Dynamics, **J. Am. Chem. Soc.**, *112*, 6127-6129 (1990)

D. Qiu, P. S. Shenkin, F. P. Hollinger and W. C. Still, The GB/SA Continuum Model for Solvation. A Fast Analytical Method for the Calculation of Approximate Born Radii, **J. Phys. Chem. A**, *101*, 3005-3014 (1997)

C. J. Cramer and D. G. Truhlar, Continuum Solvation Models: Classical and Quantum Mechanical Implementations, **Rev. Comput. Chem.**, *6*, 1-72 (1995)

G. D. Hawkins, C. J. Cramer and D. G. Truhlar, Pairwise Solute Descreening of Solute Charges from a Dielectric Medium, **Chem. Phys. Lett.**, *246*, 122-129 (1995)

M. K. Gilson and B. Honig, The Inclusion of Electrostatic Hydration Energies in Molecular Mechanics Calculations, **J. Comput.-Aided Mol. Design**, *5*, 5-20 (1991)

M. Schaefer and M. Karplus, A Comprehensive Analytical Treatment of Continuum Electrostatics, **J. Phys. Chem.**, *100*, 1578-1599 (1996)

METHODS FOR PARAMETER DETERMINATION

A. J. Pertsin and A. I. Kitaigorodsky, **The Atom-Atom Potential Method: Application to Organic Molecular Solids**, Springer-Verlag, Berlin, 1987.

D. E. Williams, Transferable Empirical Nonbonded Potential Functions, in **Crystal Cohesion and Conformational Energies**, Ed. by R. M. Metzger, Springer-Verlag, Berlin, 1981.

A. T. Hagler and S. Lifson, A Procedure for Obtaining Energy Parameters from Crystal Packing, **Acta Cryst.**, *B30*, 1336-1341 (1974)

A. T. Hagler, S. Lifson and P. Dauber, Consistent Force Field Studies of Intermolecular Forces in Hydrogen-Bonded Crystals: A Benchmark for the Objective Comparison of Alternative Force Fields, **J. Am. Chem. Soc.**, *101*, 5122-5130 (1979)

D. Hall and N. Pavitt, An Appraisal of Molecular Force Fields for the Representation of Polypeptides, **J. Comput. Chem.**, *5*, 441-450 (1984)

W. L. Jorgensen, J. D. Madura and C. J. Swenson, Optimized Intermolecular Potential Functions for Liquid Hydrocarbons, **J. Am. Chem. Soc.**, *106*, 6638-6646 (1984)

W. L. Jorgensen and 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)

J. R. Maple, U. Dinur and A. T. Hagler, Derivation of Force Fields for Molecular Mechanics and Dynamics from *ab Initio* Surfaces, ***Proc. Nat. Acad. Sci. USA***, *85*, 5350-5354 (1988)

U. Dinur and A. T. Hagler, Direct Evaluation of Nonbonding Interactions from ab Initio Calculations, ***J. Am. Chem. Soc.***, *111*, 5149-5151 (1989)

ELECTROSTATIC INTERACTIONS

M. J. Dudek and J. W. Ponder, Accurate Modeling of the Intramolecular Electrostatic Energy of Proteins, ***J. Comput. Chem.***, *16*, 791-816 (1995)

U. Koch and E. Egert, An Improved Description of the Molecular Charge Density in Force Fields with Atomic Multipole Moments, ***J. Comput. Chem.***, *16*, 937-944 (1995)

C. E. Dykstra, Electrostatic Interaction Potentials in Molecular Force Fields, ***Chem. Rev.***, *93*, 2339-2353 (1993)

C. H. Faerman and S. L. Price, A Transferable Distributed Multipole Model for the Electrostatic Interactions of Peptides and Amides, ***J. Am. Chem. Soc.***, *112*, 4915-4926 (1990)

D. E. Williams, Representation of the Molecular Electrostatic Potential by Atomic Multipole and Bond Dipole Models, ***J. Comput. Chem.***, *9*, 745-763 (1988)

F. Colonna, E. Evleth and J. G. Angyan, Critical Analysis of Electric Field Modeling: Formamide, ***J. Comput. Chem.***, *13*, 1234-1245 (1992)

POLARIZATION EFFECTS

S. Kuwajima and A. Warshel, Incorporating Electric Polarizabilities in Water-Water Interaction Potentials, ***J. Phys. Chem.***, *94*, 460-466 (1990)

J. W. Caldwell and P. A. Kollman, Structure and Properties of Neat Liquids Using Nonadditive Molecular Dynamics: Water, Methanol, and N-Methylacetamide, ***J. Phys. Chem.***, *99*, 6208-6219 (1995)

D. N. Bernardo, Y. Ding, K. Kroegh-Jespersen and R. M. Levy, An Anisotropic Polarizable Water Model: Incorporation of All-Atom Polarizabilities into Molecular Mechanics Force Fields, ***J. Phys. Chem.***, *98*, 4180-4187 (1994)

SUPERPOSITION OF COORDINATE SETS

S. C. Nyburg, Some Uses of a Best Molecular Fit Routine, ***Acta Cryst.***, *B30*, 251-253 (1974)

A. D. McLachlan, Rapid Comparison of Protein Structures, ***Acta Cryst.***, *A38*, 871-873 (1982)

R. Diamond, A Note on the Rotational Superposition Problem, ***Acta Cryst.***, *A44*, 211-216 (1988)

S. J. Kearsley, An Algorithm for the Simultaneous Superposition of a Structural Series, ***J. Comput. Chem.***, *11*, 1187-1192 (1990)

LOCATION OF TRANSITION STATES

R. Czerminski and R. Elber, Reaction Path Study of Conformational Transitions and Helix Formation in a Tetrapeptide, ***Proc. Nat. Acad. Sci. USA***, *86*, 6963 (1989)

R. S. Berry, H. L. Davis and T. L. Beck, Finding Saddles on Multidemensional Potential Surfaces, ***Chem. Phys. Lett.***, *147*, 13 (1988)

K. Muller, Reaction Paths on Multidimensional Energy Hypersurfaces, ***Ang. Chem. Int. Ed. Engl.***, *19*, 1-13 (1980)

S. Bell and J. S. Crighton, Locating Transition States, ***J. Chem. Phys.***, *80*, 2464-2475 (1984)

S. Fischer and M. Karplus, Conjugate Peak Refinement: An Algorithm for Finding Reaction Paths and Accurate Transition States in Systems with Many Degrees of Freedom, ***Chem. Phys. Lett.***, *194*, 252-261 (1992)

J. E. Sinclair and R. Fletcher, A New Method of Saddle-Point Location for the Calculation of Defect Migration Energies, ***J. Phys. C***, *7*, 864-870 (1974)

R. Elber and M. Karplus, A Method for Determining Reaction Paths in Large Molecules: Application to Myoglobin, ***Chem. Phys. Lett.***, *139*, 375-380 (1987)

D. T. Nguyen and D. A. Case, On Finding Stationary States on Large-Molecule Potential Energy Surfaces, ***J. Phys. Chem.***, *89*, 4020-4026 (1985)

T. A. Halgren and W. N. Lipscomb, The Synchronous-Transit Method for Determining Reaction Pathways and Locating Molecular Transition States, ***Chem. Phys. Lett.***, *49*, 225-232 (1977)

G. T. Barkema and N. Mousseau, Event-Based Relaxation of Continuous Disordered Systems, ***Phys. Rev. Lett.***, *77*, 4358-4361 (1996)