|  |  |
| --- | --- |
| **1.** | **Notes on Special Features & Methods** |

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/≈. 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 MICROSOFT WINDOWS SYSTEMS**

TINKER executables for Microsoft PC systems should be run from the DOS or Command Prompt window available under the various versions of Windows. The TINKER executable directory should be added to your path via the autoexec.bat file or similar. If using Win2000 or XP, set the number of scrollable lines in the Command Prompt window to a very large number, so that you will be able to inspect screen output after it flies by. With Win95/98, these Command Prompt windows are only able to scroll a small number of lines (*amazing!*), 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.

Another alternative, particularly attractive to those already familiar with Linux or Unix systems, is to download the Cygwin package currently available under GPL license from the site **http://source.redhat.com/cygwin/**. The cygwin tools provide many of the GNU tools, including a bash shell window from which TINKER programs can be run.

If the distributed TINKER executables are run 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. This keyword causes the executation window to remain open after completion until the ``Enter'' key is pressed.

**USE ON APPLE MACINTOSH SYSTEMS**

The TINKER executables are best run under Mac OS X in a ``terminal'' application window where behavior is identical to that in a Linux terminal. At present the Force Field Explorer GUI for TINKER will not run on OS X since the required Java3D extensions are unavailable.

We have discontinued active support for Mac OS 9. However, the OS 9 versions of TINKER are run 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 OS9 uses a colon instead of a forward- or back-slash 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 forms of the residues, diastereotopic hydrogens, *etc*.). However, all these types lead to literally thousands of different force field parameters. In fact, there are many 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 calculations, 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.

We are now collaborating with Anders Carlsson's group in St. Louis to add his transition metal ligand field term to TINKER. Support for this additional potential functional form is already in the TINKER source code, and we plan to release the energy routines after further testing and parameterization.

**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.

**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*.).

**EWALD SUMMATION METHODS**

TINKER contains a versions of the Ewald summation technique for inclusion of long range electrostatic interactions via periodic boundaries. The particle mesh Ewald (PME) method is available for simple charge-charge potentials, while regular Ewald is provided for polarizable atomic multipole interactions. The accuracy and speed of the regular and PME calculations is dependent on several interrelated parameters. For both methods, the Ewald coefficient and real-space cutoff distance must be set to reasonable and complementary values. Additional control variables for regular Ewald are the fractional coverage and number of vectors used in reciprocal space. For PME the additional control values are the B-spline order and charge grid dimensions. Complete control over all of these parameters is available via the TINKER keyfile mechanism. By default TINKER will select a set of parameters which provide a reasonable compromise between accuracy and speed, but these should be checked and modified as necessary for each individual system.

**CONTINUUM SOLVATION MODELS**

Several alternative continuum solvation algorithms are contained within TINKER. All of these are accessed via the SOLVATE keyword and its modifiers. Two simple surface area methods are implemented: the ASP method of Eisenberg and McLachlan, and the SASA method from Scheraga's group. These methods are applicable to any of the standard TINKER force fields. Various schemes based on the generalized Born formalism are also available: the original 1990 numerical ``Onion-shell'' GB/SA method from Still's group, the 1997 analytical GB/SA method also due to Still, a pairwise descreening algorithm originally proposed by Hawkins, Cramer and Truhlar, and the analytical continuum solvation (ACE) method of Schaefer and Karplus. At present, the generalized Born methods should only be used with force fields having simple partial charge electrostatic interactions.

Some further comments are in order regarding the GB/SA-style solvation models. The ``Onion-shell'' model is provided mostly for comparison purposes. It uses an exact, analytical surface area calculation for the cavity term and the numerical scheme described in the original paper for the polarization term. This method is very slow, especially for large systems, and does not contain the contribution of the Born radii chain rule term to the first derivatives. We recommend its use only for single-point energy calculations. The other GB/SA methods (``analytical'' Still, H-C-T pairwise descreening, and ACE) use an approximate cavity term based on Born radii, and do contain fully correct derivatives including the Born radii chain rule contribution. These methods all scale in CPU time with the square of the size of the system, and can be used with minimization, molecular dynamics and large molecules.

Finally, we note that the ACE solvation model should not be used with the current version of TINKER. The algorithm is fully implemented in the source code, but parameterization is not complete. As of late 2000, parameter values are only available in the literature for use of ACE with the older CHARMM19 force field. We plan to develop values for use with more modern all-atom force fields, and these will be incorporated into TINKER sometime in the future.

**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. Ewald summation is available for inclusion of long range interactions. Calculations are implemented via a mixture of the CCP5 algorithms of W. Smith and the Applequist-Dykstra Cartesian polytensor method. At present analytical energy and Cartesian gradient code is provided.

The TINKER package allows intramolecular polarization to be treated via a version of the interaction damping scheme of Thole. To implement the Thole scheme, it is necessary to set all the **mutual-1x-scale** keywords to a value of one. The other polarization scaling keyword series, **direct-1x-scale** and **polar-1x-scale**, can be set independently to enable a wide variety of polarization models. In order to use an Applequist-style model without polarization damping, simply set the **polar-damp** keyword to zero.

**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, Shalloway's Packet Annealing and Verschelde's Effective Diffused Potential, 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 within our research 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.

**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.