TINKER

Software Tools for Molecular Design
Version 3.9 June 2001

This manual is a summary of the TINKER User's Guide that you will find on the TINKER's website: http://dasher.wustl.edu/tinker

1 Introduction

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. Rather than incorporating all the functionality in one monolithic program, TINKER provides a set of relatively small programs that interoperate to perform complex computations. The series of major programs included in the distribution system that we will use in our class perform the following core tasks:

- build protein and nucleic acid models from sequence
- energy minimization and structural optimization
- analysis of energy distribution within a structure
- molecular dynamics and stochastic dynamics

The core TINKER system is written entirely in Fortran77.

** go into the EXAMPLE subdirectory:* ***cd tinker/example***

2 Types of Input and Output Files

This section describes the basic file types used by the TINKER package. Lets 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. Some of the .xxx extension types are 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. Almost all TINKER calculations are done from .xyz format.

** edit the anion.xyz file, using vi for example:* ***vi anion.xyz*** *(to exit, type* ***:q****)*

Note: 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 a program is run on the input file sample.xyz, the output cartesian coordinates file will be written to sample.xyz_2.
In fact the output is generally written to the lowest available, previously unused version number (sample.xyz_3, sample.xyz_4, etc., as high as needed). Input file names are handled similarly. If simply sample or sample.xyz is entered as the input file name upon running a program, then the highest version of sample.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.
One quirk of this scheme is that it becomes impossible to directly use the original unversioned copy of a file if higher version numbers are present. For example, if the files sample.xyz and sample.xyz_2 both exist, then sample.xyz cannot be accessed as input by any program. If sample.xyz is entered in response to the input file name question, sample.xyz_2 (or the highest present version number) will be used as input. The only workaround is to copy or rename sample.xyz to something else, say sample.new, and use that name for the input file.

- 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.

** edit the crambin.pdb file*

Note: TINKER does not use .pdb files directly for input/output. You will need to use auxiliary programs which are provided with the system for interconverting .pdb files with the .xyz format described above (see: 3.1 Structure Manipulation Programs)

- 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. If a molecular system specific keyfile, in this case sample.key, is not present, the TINKER program will look in the same directory for a generic file named tinker.key.

- PARAMETER FILES

The TINKER package is distributed with several force field parameter sets, implementing a selection of widely used literature force fields. The potential energy parameter files 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.
We will mainly use amber and mm3pro as potential energy parameter files.

3 Programs

3.1 Structure Manipulation Programs

This section of the manual contains a brief description of some of the TINKER structure manipulation programs. Each program comes with a detailed example showing how to run it.

- XYZPDB

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

**Example:*
You are still in the EXAMPLE directory
cp anion.xyz ../source *(copy anion.xyz in the SOURCE directory)*
cd ../source *(go into the SOURCE directory which contains all executables of Tinker. Each time you run a program, make sure that all files used by the program are in your SOURCE directory)*
./xyzpdb.x *(run the executable program)*
Enter "anion" for the Cartesian Coordinate File Name and "amber" for the Potential Parameter File Name. The program creates an anion.pdb file.
ls anion.* *(to check the creation of the PDB file)*
vi anion.pdb *(and compare with the original .xyz file)*
rm anion.* *(delete the 2 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.

**Example:*
cd ../example
cp crambin.pdb ../source
cd ../source
./pdbxyz.x
Enter "crambin" for the Protein Data Bank File Name and "amber" for the Potential Parameter File Name. The program creates a crambin.xyz file and returns the different disulfide bonds formed between residues. The program also creates a crambin.seq file which contains the sequence of the protein.
ls crambin.* *(to check the creation of the .xyz and the .seq files)*
vi crambin.seq *(note the use of the three letters code)*

vi crambin.xyz *(and compare with the original PDB file)*
rm crambin.*

- 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. Atom type numbers are automatically assigned for the specified force field. The final coordinates and a sequence file are produced as the output.

**Example: building of a right-handed alpha helix built up with 20 alanine residues*
./protein.x
Choose a name to be used for Output Files (eg: 20ala).
Enter a Title.
Enter "amber" as Potential Parameter.
Enter Residue 1: Ala -57 -47 (phi=-57 and psi=-47 being the parameters for a right-handed alpha helix)
Enter Residue 2: Ala -57 -47
...and so on until residue 20. Then press "enter" to exit when the program ask you for residue 21.
Cyclize the Polypeptide Chain: No
The program returns three files: a .xyz file with the Cartesian coordinates of the molecule, a .seq file with the sequence in amino acids, and a .int file which contains an internal coordinates representation of the molecular structure. In this course, we will only use the first two files.
ls 20ala.* *(to check the creation of the three files)*
vi 20ala.seq
vi 20ala.xyz
Now we are going to use VMD to display our molecule. First use xyzpdb to create a .pdb file that VMD can read.
Run VMD.
Click on 'Molecule' (into the Main Window).
Click on 'Load From Files'

Select 'pdb only' in the browser on the left and click on 'Select pdb'.
When you have selected your 20ala.pdb file, click on 'Load Molecule'. The molecule is displaying into the OpenGL Display window.
VMD proposes many options. Here are some of them:
type 's' to access to the scan mode with your mouse
type 't' to access to the translate mode with your mouse
type 'r' to go back to the rotate mode
Click on 'Graphics' into the main windows and try the different Drawing Method.
For more information about VMD, please read the chapter 2 (Tutorials) of the VMD User's Guide that you will find on VMD's website (see the link on the class website).

- NUCLEIC

A program for automated building of nucleic acid structures. Upon interactive input of a nucleotide sequence with optional phosphate backbone angles, the program builds internal and Cartesian coordinates. Standard bond lengths and angles are used. Both DNA and RNA sequences are supported as are A-, B- and Z-form structures. Double helixes of complementary sequence can be automatically constructed via a rigid docking of individual strands.

**Example: building of a double helix of DNA with 20 G-C base pairs*
*Go back to your SOURCE directory (**cd tinker/source**)*
./nucleic.x
Choose a name to be used for Output Files (eg: 20gc).
Enter a Title.
Enter "amber" as Potential Parameter.
Chose B-Form helix for the structure.
Enter Residue 1: g (or G or gua or GUA for guanine)
...and so on until residue 20. Then press "enter" to exit when the program ask you for residue 21.
Build a double helix using complimentary bases: Yes
As for Protein, Nucleic returns three files: .xyz, .int, and .seq.
ls 20gc.* *(to check the creation of the three files)*
vi 20gc.seq *(note the two lines, one for each strand of the double helix)*
vi 20gc.xyz

Make the 20gc.pdb file with the xyzpdb program and look at your molecule with VMD.

3.2 Potential Energy Programs

This section of the manual contains a brief description of some of the TINKER potential energy programs. Each program comes with a detailed example showing how to run it.

- MINIMIZE

The MINIMIZE program performs a limited memory L-BFGS 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/. It has a relatively fast cycle time and is tolerant of poor initial structures, but converges in a slow, linear fashion near the minimum. You have to supply 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, and the new coordinates written to updated .xyz_2 files.

**Example: minimization of the protein we built up in the PROTEIN section*
./minimize.x
Cartesian Coordinate File Name: 20ala
Enter "amber" as Potential Parameter.
Rms Gradient: 0.01
The energy is given in kcal/mol. Note the difference in energy (Initial and Final Function Value).
ls 20ala.* *(to check the creation of the 20ala.xyz_2 file)*
Use VMD to compare the structure of your molecule before and after minimization.
Before converting your new file into a PDB file, rename it (in 20ala_min.xyz for example): ***mv 20ala.xyz_2 20ala_min.xyz****.*
With VMD you can load several molecules repeating the same steps. To make

the display more clear go into Graphics and choose 'Molecule' as Coloring Method for both molecules.

- DYNAMIC

The DYNAMIC program 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). 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.

**Example:*
We will use an example found in the EXAMPLE subdirectory of the TINKER distribution: ARGON Example. It performs an initial energy minimization on a periodic box containing 150 argon atoms followed by 6 picoseconds of a molecular dynamics using a modified Beeman integration algorithm and a Bersedsen thermostat.
*Go into the EXAMPLE directory and copy all argon files into your SOURCE directory.(**cp argon.* ../source**)*
In the SOURCE directory edit argon.run and follow the given routine. Note the use of an argon.key file in which several parameters are defined.
The DYNAMIC program returns energy values and several files: argon.001, ..., argon.006 which contain the Cartesian coordinates of the argon atoms at different time steps, argon.xyz_2 which corresponds to the last Cartesian coordinates file saved (argon.006 in our example), and argon.dyn which contains the information (current position, current velocity and current and previous accelerations for each atom, as well as the size and shape of the periodic box) to start a new dynamics run from the final state of the previous run.
You can use VMD to visualize the two boxes.