Tinker 8: Software Tools for Molecular Design

Joshua A Rackers, Zhi Wang, Chao Lu, Marie L Laury, Louis Lagardere, Michael Schnieders, Jean-Philip Piquemal, Pengyu Ren, Jay Ponder

Tinker 8: Software Tools for Molecular Design

Joshua A. Rackers,[1] Zhi Wang,[2] Chao Lu,[2] Marie L. Laury,[2] Louis Lagardère,[3]

Michael J. Schnieders,[4] Jean-Philip Piquemal,[3] Pengyu Ren[5] and Jay W. Ponder[1,2]*

[1] Program in Computational & Molecular Biophysics, Washington University School of Medicine, Saint Louis, Missouri 63110, United States

[2] Department of Chemistry, Washington University in Saint Louis, Saint Louis, Missouri 63130, United States

[3] Laboratoire de Chimie Théorique, Sorbonne Universités, UPMC Paris 06, UMR 7616, case courrier 137, 4 place Jussieu, F-75005, Paris, France

[4] Department of Biomedical Engineering, The University of Iowa, Iowa City, IA 52242, United States

[5] Department of Biomedical Engineering, The University of Texas at Austin, Austin, Texas 78712, United States

*Corresponding author: ponder@dasher.wustl.edu

Abstract

The Tinker software, currently released as version 8, is a modular molecular mechanics and dynamics package written primarily in a standard, easily portable dialect of Fortran 95 with OpenMP extensions. It supports a wide variety of force fields, including the modern polarizable atomic multipole-based AMOEBA model. The package runs on Linux, macOS and Windows systems. In addition to canonical Tinker there are branches, Tinker-HP and Tinker-OpenMM, designed for use on MPI-parallel distributed memory supercomputers and on state-of-the-art graphical processing units (GPUs), respectively. The Tinker suite also includes a tightly integrated Java-based graphical user interface called Force Field Explorer (FFE), which provides molecular visualization capabilities as well as the ability to launch and control Tinker calculations.

1. Introduction

The Tinker molecular modeling package represents a complete set of software tools for performing a wide range of classical molecular simulations, with special emphasis on biomolecular calculations. This article provides an introduction to some of the features and unique capabilities of the current version of the package, Tinker 8. Recently, specialized branches of the Tinker code have become available for use on large-scale multiprocessor supercomputer systems under MPI (Tinker-HP),[1] and for GPU-based calculations (Tinker-OpenMM).[2] Integration of these codes with the Tinker suite of programs will be briefly discussed, and additional information is available in the original publications describing both Tinker-HP and Tinker-OpenMM. All of the software is available via academic web sites[3-5] and GitHub repositories.[6, 7]

Tinker originated as a new software package implementing the MM2[8] and MM3[9] force fields of Allinger for use in conformational analysis of organic natural products.[10] An early prototype of the software was incorporated as the basis of molecular mechanics calculations in the ChemOffice software package.[11] Additional applications used this early pre-Tinker platform for the development of efficient structure optimization algorithms for large molecules[12] and for packing analysis of amino acid side chains in folded protein structures.[13] Development under the Tinker name began in earnest at Washington University in the mid-1990's and the first distributed version, Tinker 3.2, was publicly announced and made available in late 1996. A major purpose of the software was, and still is, to provide a modular framework for incorporating existing empirical potentials, as well as design and parameterization of new classical force field models. More recently, Tinker served as the computational engine for the early protein folding simulations done via the Folding@home platform,[14] especially for calculations utilizing implicit

solvent models. The Tinker package and its corresponding file formats are interoperable with a variety of molecular modeling and visualization tools, including VMD,[15] PyMol,[16] Jmol,[17] Force Field X,[18] Open Babel,[19] MDTraj,[20] MDAnalysis,[21] ParmEd,[22] Molden,[23] VEGA ZZ,[24] PACKMOL,[25] ForceBalance,[26] WebMO,[27] and many others. Access to Tinker, including the AMOEBA polarizable multipole force field, is also available from the CHARMM modeling software via the MSCALE interface facility.[28]

The current Tinker 8 package contains roughly sixty command line programs written in an extended version of Fortran 95, utilizing dynamic memory allocation and OpenMP directives that enable multiprocessing across CPU cores/threads on a shared memory computer system. Figure 1 classifies the individual Tinker programs by basic functionality type. All floating-point computations are performed in full double precision arithmetic. The only hard limits on program size are the allowed number of total atoms and a small number of derived array allocations. The package is distributed with full source code and binary executables for Linux, macOS and Windows operating systems, and dimensioned for a maximum of one million atoms. Systems containing over 20 million atoms have been calculated after rebuilding, and the size is limited only by available memory. The package is designed to enable interactive use via a terminal window, or as background processes controlled via a high-level scripting mechanism. The design goal for the canonical Tinker software is to provide a transparent, modular code base that is easily and directly useable by a broad range of researchers, but efficient enough for application in many production settings.

In contrast, both Tinker-OpenMM and Tinker-HP are intended to be highly efficient computational engines on their target compute platforms, while maintaining compatibility with canonical Tinker through common coding style, algorithms, file types and general workflows.

The Tinker-OpenMM package consists of a branch of the Stanford OpenMM[29, 30] library with substantial modifications to the AMOEBA plugin, as well as an interface module written in C++ that resides between canonical Tinker and the OpenMM API. It provides a *dynamic_omm* program that exchanges data between CPU and GPU memory through the library interface and performs molecular dynamics simulations on CUDA-compatible NVIDIA GPUs. Tinker-OpenMM supports an increasing subset of Tinker's energy functions, molecular dynamics integrators, free energy methods, and other features. The current version adds an internal virial implementation for use with barostat techniques, pairwise van der Waals parameters, and is capable of running absolute and relative alchemical calculations with dual topology methods.[2] Tinker-HP is a new Tinker-compatible MPI-based, massively parallel code for molecular dynamics with an efficient domain decomposition algorithm and analytical polarization solvers. As detailed elsewhere, Tinker-HP is highly scalable across large distributed computer systems containing thousands of nodes and molecular systems containing millions of atoms.[1]

2. Features and Organization

File Types and Coordinate Representations

The Tinker files describing a particular molecular system consist of a base name followed by a suffix of three or more characters, *e.g., molecule.xyz*. Several other file name suffixes are used for various types of output, program control, *etc*. The most common default Tinker file names are listed in Table 1.

Systems are represented in Tinker as collections of points in space, typically denoting individual atoms or coarse-grained collections of atoms. File representations can contain Cartesian coordinates (*.xyz* files), full internal coordinates (*.int* files), torsional angle coordinates or rigid body coordinates. Values are stored in Angstroms and degrees, and output to a precision

of 6, 8 or 10 decimal places. Periodic box boundaries are specified in terms of crystallographic lattice lengths (*a*, *b* and *c*) and lattice angles (alpha, beta and gamma). These periodic dimensions are stored as part of the keyword control (*.key*) file for a calculation or, optionally, as part of the coordinates file itself. Periodic systems, including truncated octahedra, are defined such that the centroid of the box is located at the (0,0,0) coordinate origin.

Software Organization

The majority of the source code of the Tinker package is written in portable Fortran 95 with OpenMP parallelization directives for CPU intensive calculations on shared-memory multiple core systems. The systemwide resources are managed in Fortran modules that make use of dynamic memory allocation and are designed to only represent the current state of the simulation system. The energy specific parameters, *e.g.* the cubic and quartic coefficients of the fourth-order anharmonic bond potential, are not hard-coded in the source files, thus preserving the flexibility of Tinker in force field development.

The central component of the Tinker package is a modular set of callable routines which (1) manage the package-owned resources, including default initialization, allocation of the dynamic memory, and release of the allocated space, *etc.*, (2) perform molecular mechanical calculations and dynamics simulation on a single set of parameters and atomic coordinates, (3) read in settings from standard input, command line arguments, external files and write out the current state of the system to standard output or external files. These routines essentially work as the underlying application programming interface (API) to build the higher-level routines and programs in the Tinker package. For example, the *gradient* routine is not only called in multiple integrators but also by various minimization procedures. This design makes creating new routines and new programs easy. A good implementation example is the RESPA integrator. For

this integrator the energy and force terms are organized into "fast" and "slow" groups, evaluated on different time scales. Because these energy and force routines are organized as a callable library, RESPA is integrated at a high level by simply toggling these terms on and off.

Keyword Control Mechanism

Every program in the Tinker package is capable of interactively reading arguments from standard input, thus making the program easy to use directly. These interactive inputs are limited to the basic necessities for any given calculation. However, the Tinker programs are not restricted to reading runtime arguments from the command line. Advanced users can set more detailed options via an external configuration (*.key*) file through a "keyword" mechanism. The keywords not only manipulate the straightforward behavior of the programs, (*e.g.* whether or not to save the velocities of atoms during a simulation), but also manage default settings (*e.g.* to change the grid dimension used by PME, as necessary), handle hardware resources (*e.g.* setting a number of threads for OpenMP, choosing an available GPU card, *etc.*), and even control library dependency (*e.g.* switching between underlying FFT algorithms). The current Tinker version implements about 350 keywords, many with multiple options to provide fine-grained control over the behavior of Tinker calculations.

How to Write a Tinker Program

Tinker has an intentionally modular design. In addition to making the code easily understandable, this modularity makes it possible to quickly write new Tinker programs. For most applications, a new program can be initialized, a structure read in, and a molecular mechanics model set up in just three lines of code:

```
call initial
call getxyz
call mechanic
```

These steps, shown in more detail in figure 2 allow developers to use Tinker's existing machinery to quickly set up new types of calculations.

The first step in writing any new Tinker program is initialization of variables and reading of a molecular structure. If the new program doesn't require any new global variables, this can be done via the *initial* and *getxyz* routines. *Initial* declares and initializes global variable values that are needed for every Tinker program. *Getxyz* parses a Tinker Cartesian coordinates file (*.xyz*) for a molecular system, provided either via command line input or interactively at a user prompt. Once these two routines are called, Tinker is ready to perform operations on the structure. Multi-structure "trajectories" can also be read directly as input from Tinker archive (*.arc*) files.

Once a structure is obtained, the work of setting up a Tinker molecular mechanics calculation is performed by the *mechanic* routine, which is a self-contained protocol for setting up the potential energy model for a given system. First it assigns connectivity to the structure and obtains a force field parameter file (*.prm* file). This can be supplied at an interactive prompt, or included in a keyword control file (*i.e.*, a "keyfile", typically *.key*) containing Tinker directives or "keywords". Then *mechanic* does all the work of setting up the potential energy function. If no keyfile is supplied, the package simply instantiates the contents of the parameter file. If a keyfile is provided, it may optionally contain keywords related to each individual component of the potential energy function and specifying modified or additional parameter values that supersede those in the parameter file. The internal setup for each potential energy term is also highly standardized. For example, the multipole energy, force and Hessian routines, all of which have source files named *empole**, have a corresponding initialization routine named *kmpole* that assigns force field parameters to atoms or groups within the molecular structure. There is a corresponding "*k*" routine for every potential energy component included in Tinker. Adding an

entirely new potential energy function is also straightforward. The developer simply adds the code for the function to the preexisting, empty *extra* energy and force routines, which have full access to the molecular data structures, and then edits *kextra* to read in any new parameters or keywords that might be needed for the new potential. Tinker is then set to utilize these routines automatically, and to optionally include them in a force field model.

Providing the tools to easily read in structures and construct models minimizes the work of setting up and debugging Tinker data structures and eases the development of new methods. This modularity, particularly of the potential energy functions, allows developers to quickly alter components of calculations without having to make changes across multiple files. It provides developers the opportunity to create in their own new potential energy terms, force field parameters and keyword control features without having to navigate a maze of source code.

3. Computational Models

Potential Energy Functions

Among the many goals of the Tinker software package, one of the most fundamental is to allow users the ability to explore a wide variety of models. Regardless of whether a user is using new Tinker program they have written or an existing one, every energy-based calculation requires definition of a force field model. To this end, Tinker includes support for a tremendous array of potentials. There are two advantages to the large number of potentials that are included and supported by the package. First, it gives end users the ability to use and compare a wide variety of models for their particular application system. To this end, various Tinker potential terms can be grouped together to replicate several widely used biomolecular force fields such as those from the CHARMM,[31] Amber[32] and OPLS-AA[33] families. The second reason to support a large number of potentials is to expedite the development of new models. Because of the

modular nature of the code, researchers can easily incorporate any of the existing potentials in a model. In total there are approximately thirty different potential terms supported in the Tinker package, all with exact analytical energies and Cartesian derivatives, and many with second derivatives. Broadly, the potentials can be divided into intramolecular terms, intermolecular terms and implicit solvent models.

The intramolecular potential energy terms in Tinker can be further subdivided into primary terms and cross terms. The former describe the energetics of simple motions such as bond stretching, angle bending and torsional rotation, while the latter describe coupling between the primary energy terms. The simplest of the primary terms are the bonded potentials. Tinker includes harmonic, anharmonic and Morse bond potentials. The package also has several types of angle bending potentials – harmonic, anharmonic, linear, projected in-plane and Fourier-based angles. Additionally, there are four types of torsion terms included in Tinker. The first is a calculation for a simple torsion defined by four consecutively bonded atoms using a sum of Fourier terms. The second, referred to as a Bell's "pi-torsion", computes the torsion around a bond connecting two trigonal centers using the pi-orbital directions at each trigonal center.[34] Tinker also includes so-called "improper torsion" terms that define torsionals between non-consecutively bonded atoms, as used to enforce planarity in the Amber models and many other force fields. Finally, harmonic "improper dihedral" terms can be used to maintain planarity, as in the CHARMM force fields. An additional primary potential term is the direct description of out-of-plane bending. Tinker has thee methods for computing an out-of-plane bending potential. The first two potentials are computed via an out-of-plane angle, using either the Wilson-Decius-Cross[35] or Allinger[36] definitions. A simpler, third method consists of a harmonic term describing the out-of-plane distance of a trigonal atom from the plane defined by its three attached atoms.

These primary terms describing the energetics of bonds, angles, torsions, and out-of-plane bends comprise the bulk of most intramolecular energy models a user might like to build or use.

In addition to primary intramolecular potentials, Tinker also supports a variety of intramolecular cross terms. These terms control how the primary energy models are coupled and change as a function of each other. The classic and most basic example of a cross term is the stretch-bend or bond-angle term, which describes how two adjacent ideal bond distances change as a function of the angle between the bonds. Included in Tinker, in addition to a stretch-bend potential, are cross terms for angle-angle, bond-torsion, angle-torsion and torsion-torsion terms as well as a Urey-Bradley[37] term. Including these terms in a total potential allows users to build and use sophisticated intramolecular energy models when the application requires it, for example to reproduce vibrational frequencies.

The next broad class of potentials provided by Tinker are intermolecular terms. These can be subdivided into van der Waals (vdW) or repulsion-dispersion interactions, and generalized Coulombic or electrostatic interactions. In order to support a wide variety of models, Tinker includes five different functional forms for van der Waals interactions: a Lennard-Jones 6-12 potential[38], buffered 14-7 Halgren potential,[39] Buckingham exponential-6 potential,[40] a Gaussian vdW potential, and the MM3 vdW-hydrogen bond potential.[41, 42] These functions allow a great deal of flexibility in using and designing models with different representations of short-range interactions between atoms.

The most complex set of potentials included in the Tinker package are the electrostatic interaction potentials. Tinker has the ability to compute simple point charge interactions, but it also implements interactions between higher-order multipole moments. Tinker can treat bond-center dipole models, permanent atomic multipole models with interactions through quadrupoles,

and induced dipole models. The ability to efficiently compute permanent multipole and induced dipole models allows Tinker to run calculations with more advanced models, such as the AMOEBA force field.[43-47] Indeed, much development effort in Tinker has been and continues to be focused on streamlining and modularizing code to implement next-generation force fields with more accurate electrostatic models.

The last major category of potentials in Tinker is continuum models. The most commonly used of these are various implicit solvation models. Tinker includes support for several Generalized Born (GB)[42] variations including those of Still,[43] Onufriev-Bashford-Case,[48] ACE[49] and Grycuk,[50] the Generalized Kirkwood (GK)[51] method for use with polarizable multipoles, accessible surface area-based solvation,[52] the Hydrophobic Potential of Mean Force (HPMF),[53] a novel reaction field method,[54] and Poisson-Boltzmann (PB)[55-57] solvation models. The GB, GK, surface area and HPMF potentials are all implemented directly in the Tinker code while PB calculations are provided via an interface to the Adaptive Poisson-Boltzmann Solver (APBS) software package.[58, 59] All of the solvation models in Tinker are implemented to work with advanced electrostatic and induced dipole models. In addition to these solvation models, Tinker also includes surface area and volume calculations with derivatives, which can be used to build or use potentials incorporating these geometric molecular descriptors.

Additionally, Tinker includes two orbital-based models for description of select quantum effects within a classical framework. Simple pi-orbital calculations of the Hückel, Pariser-Parr-Pople, or variable electronegativity self-consistent field (VESCF)[60] class can be used to scale bond and torsional parameters in conjugated or aromatic systems. Three ligand field models for describing the coordination geometry at transition metal sites within the Tinker package have also been described.[61-63]

Although Tinker includes a large number of possible potentials, using them within an energy model is straightforward. The energy and gradient subroutines for each different potential are modular, which is to say they can each be called separately with just one line of computer code. For developers this means it is easy to mix-and-match different potentials in a model or devise new potential as desired. For users this makes it simple way to activate or deactivate individual parts of a model via a single keyword to toggle use of individual potential terms. This makes it easy to manipulate and analyze energy components for complicated structures.

Force Field Models

The wide variety of classical functional forms available in Tinker enables support of a number of existing force fields. From its beginnings Tinker has been intended for use with multiple models. In fact, one of the original goals of the package was to allow users to seamlessly compare energetic models for a given problem or application. To this end Tinker supports the following standard force fields: Amber,[64-68] CHARMM,[69-72] OPLS,[73-78] MM2/3,[8, 79-83] MMFF,[84] AMOEBA,[44-46, 85-87] Dang,[88-97] the so-called "Tiny" force field, and a number of specialized models for water. For many of these force fields, several modifications are provided as complete parameter sets contained within the Tinker distribution.

The force fields available in Tinker span a wide range: from the Tiny force field with generic parameters based on element type and valence for use in optimizing crude structures to the AMOEBA09 small molecule force field containing detailed parameters over finely subdivided atom types and advanced functional forms such as multipolar electrostatics and induced dipole polarizability. The included force fields also span major classes of biomolecules, with parameters to model proteins, nucleic acids, lipids, and small organic molecules. Users

should consult the respective literature on each force field before deciding which model might be best suited to their application.

4. Capabilities

Structure Manipulation

In order to generate coordinate files adapted to various software packages and purposes, Tinker provides convenient tools to convert coordinate files into different formats and to manipulate the coordinate file for different calculation purposes, such as building crystal structures, generating periodic boxes etc.

First, Tinker recognizes the Tinker .xyz file format for all calculations. However, other software packages are adapted to coordinate files of other formats. For instance, CHARMM, AMBER and VMD are adapted to PDB files, SYBYL are adapted to MOL2 files and many QM packages such as Gaussian is able to read in internal coordinate files. To allow interoperability, Tinker provides six commands to do the interconversion between different coordinate files. The command *pdbxyz* takes a Tinker xyz file as input and generates the corresponding PDB file as output. The command *xyzmol2* converts a Tinker xyz file to a MOL2 file. The command *xyzint* converts an xyz file to an internal coordinate file in which the absolute Cartesian coordinates are expressed as relative positions (bond length, bond angle and torsional angle) among atoms. The commands *pdbxyz, mol2xyz,* and *intxyz* convert PDB files, MOL2 files and internal coordinate files back to xyz files.

Second, Tinker also provides file-editing tools for the purpose of simulation setup. Most of xyz editing tools are listed as options under the command *xyzedit,* such as inserting and deleting atoms, changing force field atom types, translating/rotating a system to specified Cartesian or rigid body coordinates or into the inertial frame, appending and merging multiple

files, or soaking a second xyz file, creating a periodic boundary box, placing a solute into a periodic solvent box, adding ions to a solvated system, *etc*. The command *superpose* is designed to superimpose a pair of structures to at optimal root mean square deviation (RMSD) using a non-iterative quaternion-based algorithm.[98] Since biomolecules such as nucleic acids and proteins are target systems for many studies, Tinker provides *nucleic* and *protein* tools to generate nucleic acid and protein structures respectively according to the sequence information and backbone or side chain torsional angle values. Lastly, the utility *crystal* utility is designed for manipulation of crystal structures such including generation unit cells from asymmetric units and according to box size, shape and space group.

Local Search and Minimization

Tinker has a number of local minimization algorithms implemented to effectively and efficiently minimize a quantity of interest. Several algorithms are widely used in Tinker in conjunction with a force field to minimize the energy of a molecular structure. The code contains routines for Limited Memory Broyden–Fletcher–Goldfarb–Shanno (LBFGS)[99-101] minimization, Optimally Conditioned Variable Metric (OCVM)[102, 103] nonlinear optimization, and Truncated Newton Conjugate Gradient (TNCG)[12, 104] Hessian-based optimization. The LBFGS algorithm is of the nonlinear conjugate gradient class, and as such does not require an analytical Hessian matrix. It uses the BFGS update to update the line search direction at each iteration. The limited memory implementation in Tinker allows this routine to be used for Cartesian minimization of large systems. The OCVM algorithm uses a quasi-Newton methodology without line search to update an approximation to the inverse Hessian at every step. It is particularly effective for optimization of rougher potential surfaces, such as those in torsional space. Lastly, the TNCG algorithm uses a preconditioned truncated conjugate gradient method coupled with direct sparse

Hessian evaluation or a finite-difference Hessian approximation to minimize an objective function. The TNCG method converges quadratically once in the vicinity of a local minimum and can optionally find transition states and general stationary points after disabling checks for negative curvature. LBFGS and TNCG use the same line search algorithm, a gradient-based trust region safeguarded parabolic extrapolation, cubic interpolation procedure. To minimize structures, the LBFGS, OCVM and TNCG methods are implemented in the Tinker *minimize*, *optimize* and *newton* programs, respectively. These minimize structures in Cartesian coordinate space. Tinker also contains the corresponding programs, *minirot*, *optirot* and *newtrot* for minimizations in torsional space as well as *minrigid* and *optrigid* for minimizations with rigid body groups of atoms.

While TNCG-based optimization methods are easily modified to allow convergence to transition states, the catchment basin is often small and requires a starting structure near to the final transition state. Tinker contains two other methods, *saddle* and *path*, that are specifically designed to locate conformational transition states and pathways. *Saddle* represents a combination of ideas from the Halgren-Lipscomb synchronous transit[105, 106] and Bell-Crighton quadratic path[107] methods. It takes two endpoint structures as input, and performs an iterative series of maximizations along the connecting path and minimizations orthogonal to the path until the saddle point is located. The *path* program starts from local minima and uses Lagrange multiplier-based constraints to minimize orthogonal to a series of equally spaced path points, generating a "trajectory" along the interconversion pathway.[108]

In addition, Tinker contains an adaptive derivative-free multi-dimensional Nelder-Mead simplex optimization algorithm and a modified Levenberg-Marquardt least squares algorithm combining features of the IMSL BCLSF routine and the LMDER code from Minpack.[109] These

methods are used within Tinker for optimization of stochastic objective functions and in force field parameter refinement, respectively.

Global Optimization

Besides the various optimization methods to find local minima of potential energy functions, Tinker also has a number of optimization algorithms to find global minima of the target function. Roughly, these algorithms can be divided into two categories: first, methods that rely on pathway- or trajectory-dependent propagation to overcome the local barriers or to enumerate local minima; second, methods that modify the underlying potential surface while approximating a solution to the equilibrium density distribution. The first category of methods includes simulated annealing,[110] generalized gradient descent,[111, 112] "Jumping-between-Wells"[113] and the Monte Carlo minimization (MCM) method.[114, 115] The second category of global optimization algorithms includes potential smoothing techniques [116-119] and the related gaussian density annealing (GDA) scheme.[120]

The *anneal* program is a traditional MD-based simulated annealing code with an optional pre-equilibration phase and several available cooling schedules. It starts from a high temperature at which local energy barriers are easily to overcome. Then the cooling schedule is applied to gradually lower the temperature and coalesce into a low energy local minimum. In the *sniffer* program, a second order differential equation is designed to enable generalized descent along a trajectory without becoming trapped in the catchment region of any particular minimum. Following a steepest descent propagator, the trajectory is constrained to a minimum that is greater than the predefined energy levels, which is presumed to be the global minimum.[111, 112, 121] The *scan* program uses Jumping-between-Wells to locate all the local minima for an input structure by self-consistently following low frequency normal mode search directions from all

known minima. The global minimum can be obtained by comparing all the local minima.[113] The *monte* program implements an MCM protocol which uses Metropolis Monte Carlo exploration of a potential surface where the energy of each point on the surface is remapped to the value of the closest local minimum.[114] Potential surface smoothing (PSS) views the original potential energy functional forms as the time zero initial conditions for solution of the diffusion equation. Conformational search is then performed on the smoother surface produced at some finite, non-zero time. The method can be shown to be mathematically equivalent to performing molecular mechanics with "fuzzy" atoms, where the location of each atom is generalized to a Gaussian probability distribution around its most likely position. The *pss*, *pssrot* and *pssrgd* programs implement the PSS idea in terms of Cartesian, torsional and rigid-body representations, respectively. The *gda* program performs annealing while seeking an approximate solution for the equilibrium density distribution, and can be viewed as a dynamical version of the deterministic potential smoothing methods.

Two examples of the global optimization methods are demonstrated in figure 4 for a gas phase deca-alanine model system in gas phase using the *scan* and *monte* programs. Both optimizations start from the same linear structure of Deca-Alanine and eventually reach the same global minimum, the structure of which is a typical α-helix as shown in figure 4A and 4B. The *scan* method captured 654 intermediate structures while scanning the full potential surface. The *monte* method generated eight intermediate local minima along its path to the helical structure. Two intermediate structures from each calculation are presented in figure 4A and 4B. Though they follow different paths in moving around the surface, both methods appear to produce find a similar partially optimized structure, shown as intermediate minimum II in figure 4.

Dynamics Methods

One of the important features for any modern molecular mechanics software package is the ability to perform molecular dynamics (MD). In the past four decades, many of the important contributions of classical empirical potential models have been realized through MD simulations. In Tinker this feature is implemented through the *dynamic* program: a feature-rich MD engine. In addition to being able to run simulations with any of the force fields included with Tinker, it allows the user a great deal of flexibility in the details of how a simulation is run.

Tinker has the ability to run simulations in any of four traditional statistical mechanical ensembles: Microcannonical (NVE), Cannonical (NVT), Isoenthalpic-Isobaric (NPH) and Isothermal-Isobaric (NPT). For each of these options, where necessary, Tinker can employ a wide variety of integrators, thermostats and barostats. The possible integrators include Velocity Verlet, Beeman,[122, 123] stochastic,[124, 125] Nosé-Hoover NPT,[126] Bussi-Parrinello NPT,[127] a two-stage, multiple time step, reversible Reference System Propagator Algorithm (RESPA)[128, 129] and a rigid-body integrator.[130] Most of these integrators have been reviewed extensively in the literature. Two of particular interest, however are the RESPA integrator and the rigid body integrator. The RESPA integrator allows the user to take two separate time steps when propagating molecular dynamics. The first, fast time step is used for fast-changing degrees of freedom such as bond stretching and the second, slow time step is used for the slow-changing, but computationally expensive electrostatics or polarization calculations. The rigid body integrator is unique to Tinker and is based on the original work of Andrey Kutapov and Marina A. Vorobieva (VNIITF, Russian Federal Nuclear Facility, Chelyabinsk). Tinker also includes an implementation of the RATTLE algorithm[131] in order to implement holonomic constraints within velocity Verlet and related integrators. In addition, Tinker contains a stochastic dynamics

integrator[132] employing a series expansion to treat small frictional coefficients,[133] and with the ability to scale the friction term based on accessible surface area.[134]

For the constant temperature and pressure ensembles, Tinker includes a variety of thermostats and barostats. The included thermostats are Bussi[135], Berendsen[136], Andersen[137] and Nose-Hoover.[126, 138] The available barostats are Berendsen,[136] Bussi-Parrinello[127] and Monte Carlo.[139] It should be noted that because Tinker includes an internal virial calculation for every available model potential, the Berendsen barostat may be used with both simple and advanced models. The defaults in Tinker are the Bussi thermostat and Berendsen barostat, but the available thermostats or barostats can be used in any of several combinations with the standard integrators (Verlet, Beeman and RESPA). An active area of development in Tinker is application of an isokinetic scheme that combines a massive thermostat with a multiple time step integrator to achieve ultra-long time steps for the slowly evolving, but computationally expensive, potential terms in a simulation. This method is deemed Stochastic-Iso-NH-RESPA or SIN(R) and it has been demonstrated to achieve outer time steps of up to 100 fs for the AMOEBA water model without loss of model accuracy.[140, 141]

Properties and Analysis

One of most useful programs in the Tinker package is *analyze*. It can be used to evaluate a single structure or a multiple-frame file from a simulation. The program is designed to provide everything from general information to detailed atom-level information about the system. Its most basic function is to simply print out the total potential energy broken down into each individual component but can do much more. The *analyze* program can give information about the force field being used and the parameters for every atom in the system. It optionally outputs a potential energy breakdown by atom, or with details for every interatomic interaction. It can also

give the user some basic properties of the system, such as electric moments and principle axes. *Analyze* calculates the internal virial, numerical and virial-based derivatives of the energy with respect to volume. And lastly it can print the connectivity list and force field parameters used for every atom and interaction. As with many Tinker programs, analyze can take as input either a single structure as an *.xyz* file, or a multi-frame archive or MD trajectory as a Tinker *.arc* file. These features not only allow users to evaluate properties for single structures or trajectories, but also to quickly spot and isolate any errors or inconsistencies that might occur.

Tinker implements analytical Hessian computation for many potential functions, and numerical Hessian evaluation for all others. The Hessian is arranged in a sparse matrix with only elements with magnitude greater than a keyword specified cutoff stored. The *vibrate* program finds the mass-weighted Hessian, and after diagonalization via the *diagq* routine (Bernard R. Brooks, NHLBI, NIH), produces the normal modes and vibrational frequencies for the input structure. Small multi-frame structure files are also generated to enable visualization of the motion along each mode.

For large structures, such as biopolymers, where full matrix diagonalization is not practical, the vibbig program implements an iterative sliding block diagonalization method that finds the lowest frequencies and corresponding modes with $O(N^2)$ computational effort.[142]

In addition to analysis and manipulation of structures, Tinker has a suite of programs designed to assess properties for liquid systems. The *diffuse* program takes as input an MD trajectory as a *.arc* file and calculates the self-diffusion coefficient of a homogeneous liquid or subset of atoms from a heterogeneous system. The algorithm employed uses the standard Einstein relation applied to the molecular centers of mass of the liquid. There are also programs

to compute the bulk dielectric constant and radial distribution function (*radial*) starting from an input dynamic trajectory.

Correlate as a general program and formalism for computation of time correlation functions. It has built-in methods to find structural correlation and velocity autocorrelation functions. In addition, users can provide an external routine to compute any structure- or energy-based property, and correlate will generate its correlation function. Additionally, the velocity autocorrelation function is used as input to the Tinker *spectrum* program, which computes the corresponding power spectrum. This suite of programs gives users a set of tools to assess properties from liquid simulations.

Free Energy Calculations

One of the most common applications of molecular modeling is the calculation of binding free energies. To compute the binding free energy of a drug to a protein or solvation free energy of an ion in water, Tinker has methods available. Computation of binding free energies relies on the completion of a thermodynamic cycle, as pictured in figure 5. In order to calculate a free energy, Tinker employs an "alchemical" approach that "disappears" the ligand of interest in the presence and absence of its host. The free energy differences of these processes are calculated using free energy perturbation.

The majority of the analysis of the free energy difference of the sampled conformations in Tinker package is handled by the *bar* program. *Bar* applies the standard Zwanzig's free energy perturbation (FEP) method[143] and Bennett's acceptance ratio (BAR) method[144] for the canonical ensemble. Additionally, the *bar* program has been extended to process isothermal-isobaric simulations[145] and to estimate the differences in entropy and enthalpy of the samples.[146]

An example of the utility of the *dynamic* and *bar* programs is calculation of binding free energies for the SAMPL4 host-guest challenge.[147] We used *dynamic* to run sampling simulations of the host-guest binding systems over λ-windows to decouple guest electrostatic and van der Waals interactions, and then performed *bar* free energy perturbation calculations on those trajectories. The results for one particular host-guest pair are shown in figure 3. In addition to prediction of the binding free energy, *dynamic* trajectory snapshots show the preferred binding pose for this ligand.

Testing and Debugging

All of the analysis procedures listed above depend on the validity of the model that goes in to them. Tinker has many built-in utilities to test the correctness of code for new existing and new models. These allow developers to quickly test if a new energy function and its derivatives are consistent. The *testgrad* and *testrot* programs check to make sure the analytical potential energy derivatives match those calculated numerically. *Testgrad* operates in Cartesian space, while *testrot* computes and checks derivatives with respect to torsional angles. The *testhess* program takes this the next step by comparing the analytical Hessian against one computed numerically from either gradient or energy values. It can calculate the numerical hessian from either the potential energy or the gradient. Finally, the *testpair* utility tests methods for determining pairwise neighbor interactions in energy and gradient evaluation. This program compares results and computes timings for energy and gradient evaluations using a double loop, the method of lights or a pairwise neighbor list.

In addition, Tinker includes *polarize*, a program to compute the molecular polarizability of an individual molecule using either an additive or interactive induced dipole model. In

addition to being able to compare with experiment values, computing molecular polarizability gives users an idea of how strongly many-body effects may affect subsequent calculations.

Parameterization Tools

The final set of important utilities in Tinker are a trio of programs designed to parameterize new molecules. The Tinker *valence*, *poledit* and *potential* programs can be used to generate parameters for intra- and intermolecular potential energy functions. The *valence* program takes a Tinker *.xyz* file and a Gaussian QM output file and generates a set of parameters for the basic intramolecular potential energy function as well as rough guesses at van der Waals parameters. It can also further refine those intramolecular energy function parameters by fitting to QM calculation results. The *poledit* program allows users to set and modify atomic multipole models. It can generate multipole parameters obtained from Gaussian Distributed Multipole Analysis (GDMA) output.[148] It is also used to set local coordinate frames for atomic multipole, modify polarizability values, define polarization groups for the AMOEBA model, and average multipole parameters for symmetry-related sites.

Lastly, the *potential* program can be used for the evaluation and refinement of atomic multipole models. This utility computes the electrostatic potential on a grid of point surrounding a molecule. It can then either compare that potential to another multipole model or QM calculation or fit the multipole model to the QM result. These three parameterization programs are combined in a Python-based, publicly available software package called Poltype.[149] This program is specifically designed to automate the process of generating parameters for the AMOEBA model and has been used extensively to facilitate rapid and reproducible parameterization of new molecules.

5. Algorithms

One of the challenges faced by all molecular modeling packages is efficient calculation on large application systems. Tinker incorporates a number of interesting and novel algorithms to help address computational bottlenecks, including algorithms of for periodic boundary calculations, neighbor list generation, particle mesh Ewald summation for electrostatics, and efficient induced dipole solvers for polarization.

Periodic Systems and Neighbor Lists

To enable modeling of "infinite" systems, four types of periodic box are supported in Tinker. These are orthogonal, monoclinic, triclinic and octahedral, where the octahedral periodic box refers to a truncated octahedron derived from the corresponding cube. When the cutoff of the periodic boundary condition is so large that the neighbors of an atom include at least two images of the same atom, a unique "replica" method is enabled automatically to replicate the periodic box to account for this situation. Tinker provides four internally built neighbor lists whose cutoff distances and list buffers can be configured separately through keywords for the van der Waals, the partial charges, the atomic multipoles and the polarization preconditioner, respectively, to speed neighbor searching as opposed to the naïve double loop method only if the replica method is not enabled. An efficient, OpenMP parallel neighbor list updating mechanism is used to minimize list rebuilding overhead. The *Method of Lights*[150] can be used to efficiently construct the neighbor lists for the triclinic, monoclinic and orthogonal boxes. Finally, the periodicity code in Tinker is able to handle infinite bonded polymers by tracking valence terms across periodic cell boundaries. This enables correct treatment of the diamond lattice, rubber, graphite, plastics, and similar large repeating systems.

Particle Mesh Ewald Summation

To speed electrostatics and polarization calculations on large systems, Tinker has the ability to use smooth particle mesh Ewald summation (PME) for models including charges, multipoles or induced dipoles. Descended from an original PME code written by Thomas Darden, Tinker 8 gives the user control over the Ewald damping parameter and the allows use of either "tinfoil" or vacuum boundary conditions. The PME module also supports truncated octahedra as a periodic shape and allows performing PME calculations on a non-periodic systems. The current Tinker implementation follows closely the multipole PME version previously described by Sagui, *et al.*[151] The code follows the structure of typical PME software: putting the electrostatic moments onto a spatial grid, performing a Fourier transform, performing the potential and electric field calculations in Fourier space, transforming back to real space, and finally computing the energy and force on every atom. One unique feature of the code is a domain decomposition scheme for putting moments on the grid. This method, developed by David Gohara (Biochemistry, Saint Louis University), parallelizes this step, which otherwise is the rate limiting computational step for large systems. Tinker optionally uses either a refactored 3D version of the public domain FFTPACK Fourier transform code, or the Fast Fourier Transform package FFTW (Fastest Fourier Transform in the West)[152] to perform the forward and backward Fourier transforms necessary for PME calculations.

Polarization Algorithms

One of the defining features of Tinker is its ability to run simulations with force fields that include induced dipole polarization. The foundational idea of such models is that the induced dipole at a given site is proportional to the electric field at that site according to

$$\vec{\mu}_i = \alpha_i \vec{F}_i$$

where μ, α and F represent the induced dipole, the polarizability, and the electric field respectively. In a mutually inducible model, the electric field arises not only from the permanent moments of the systems, but the induced dipoles as well.

$$\vec{F}_i = \vec{F}_i^{perm} + \vec{F}_i^{ind}$$

This gives rise to the total induction energy,

$$U^{ind} = \frac{1}{2}\sum_i \vec{\mu}_i \bullet \vec{F}_i^{ind} - \sum_i \vec{\mu}_i \bullet \vec{F}_i^{perm}$$

where all that is needed is to solve for the induced dipoles of the system. Tinker has three methods of determining the induced dipoles of a system: Preconditioned Conjugant Gradient (PCG), Optimized Perturbation Theory (OPT) and Extended-Lagrangian/Self Consistent Field (iEL-SCF).

The most straightforward way to obtain the induced dipoles of a system is by requiring a zero residual,

$$R = \left(\frac{dU}{d\vec{\mu}}\right) = 0 .$$

which enforces that the change in energy should be zero for an infinitesimal change in the induced dipoles. Solving this system of equations is a flavor of the familiar self-consistent field (SCF) calculation. In Tinker this is done using a preconditioned conjugate gradient algorithm[153] and is typically able to converge the calculation within 5-6 iterations.

The OPT method[154, 155] works in a manner similar to PCG, but instead of iteratively lowering the residual, it computes induced dipoles from perturbation theory. In this scheme the exact induced dipoles are expanded in a power series,

$$\vec{\mu}_{tot} = \vec{\mu}_0 + \lambda\vec{\mu}_1 + \lambda^2\vec{\mu}_2 + \ldots + \lambda^n\vec{\mu}_n$$

where each order of the perturbation is determined by:

$$\begin{aligned}
&\vec{\mu}_0 = \alpha\vec{F}^{perm} \\
&\lambda\vec{\mu}_0 = \lambda\alpha\vec{F}^{ind(\mu_0)} \\
&\lambda^2\vec{\mu}_1 = \lambda^2\alpha\vec{F}^{ind(\mu_1)} \\
&\vdots \\
&\lambda^n\vec{\mu}_n = \lambda^n\alpha\vec{F}^{ind(\mu_{n-1})}
\end{aligned}.$$

In this expansion, each order of dipole determined by the one that precedes it. This gives rise to a final energy expression,

$$\begin{aligned}
U &= \sum_i \vec{\mu}_i^{OPT} \bullet \vec{F}_i^{perm} \\
\vec{\mu}^{OPT} &= M_0\vec{\mu}_0 + M_1\vec{\mu}_1 + M_2\vec{\mu}_2 + \ldots + M_n\vec{\mu}_n
\end{aligned}$$

where the M coefficients are parameters that can tuned. Tinker currently has the ability to include up to six terms in this expansion, but it has been shown that including only three is a reasonable approximation that gives a speed boost over traditional PCG.

The final method included with Tinker is the iEL-SCF method.[156] This method minimizes the number of iterations needed in solving the induced dipoles by introducing the Lagrangian,

$$L = \frac{1}{2}\sum_i m_i\dot{\vec{r}}_i^2 + \frac{1}{2}\sum_i m_{\mu,i}\dot{\vec{\mu}}_i^2 - U_{AMOEBA}(\vec{r}^N, \vec{\mu}_{SCF}^N) - \frac{1}{2}\omega^2\sum_i m_{\mu,i}\left(\vec{\mu}_{SCF,i} - \vec{\mu}_i\right)^2$$

where m_i represents the mass of atom i, $m_{\mu,i}$ a fictitious dipole mass and ω the frequency of the harmonic potential that keeps the induced dipoles close to the fully converged SCF solution. By applying Lagrangian equations of motion, one obtains the classical equation of motion plus the equation of motion for the auxiliary degrees of freedom,

$$\begin{aligned} m_i \ddot{\vec{r}}_i &= -\frac{\partial U_{AMOEBA}(\vec{r}^N, \vec{\mu}^N_{SCF})}{\partial \vec{r}_i} \\ \ddot{\vec{\mu}}_i &= \omega^2 \left(\vec{\mu}_{SCF,i} - \vec{\mu}_i\right) \end{aligned} .$$

To maintain stability, a thermostat is applied to the auxiliary degrees of freedom. This gives the iEL-SCF method the ability to reduce the number of iterations needed to obtain induced dipoles for a system and thus speed up simulations.

In addition to these methods, there are future plans to include at two additional polarization options into Tinker 8. The first is an extension of the iEL-SCF method called iEL-0SCF.[157] This method uses the same auxiliary dipoles from the iEL-SCF scheme, but instead of using them as a starting point for SCF, they are used to drive dynamics directly. By avoiding SCF iterations, the iEL-0SCF method does not produce fully converged dipoles but does allow for much faster, stable MD simulations. The second method, already incorporated into the Tinker-HP code base is the Truncated Conjugate Gradient method (TCG).[158] This approach computes a fixed number of iterations of the conjugate gradient algorithm and then corrects for the fact that the residual has not been minimized to zero. By using successive approximations from the conjugate gradient iterations this method avoids needing any parameters as are needed in the previous approximate methods listed. Moreover, by correcting for the lack of zero residual, the TCG method allows for faster computation of analytical induced dipoles than full

SCF methods like PCG. Both of these methods are slated for implementation in the next release of Tinker.

Orthogonal Space Random Walk

Besides the typical Free Energy Perturbation (FEP) method, the Orthogonal Space Random Walk (OSRW) free energy calculation method is also implemented in Tinker. Classical FEP methods (BAR, thermodynamic integration, *etc*.) arbitrarily select an order parameter to sample. The OSRW method is capable of exploring the order parameter as well as the so-called "hidden degrees of freedom" simultaneously.[159, 160] Due to the complexity of many systems, efficiently sampling the hidden degrees of freedom dominates the accuracy of final free energy computation. Currently, OSRW free energy calculations in Tinker are supported for the NVT ensemble and RESPA integrator, and are restricted to the buffered 14-7 vdW potential where a softcore-modified buffered 14-7 potential is applied as a replacement for the original. Permanent electrostatic interactions are also modified by a softcore treatment, to prevent numerical instability during simulation.[161] When using OSRW with AMOEBA, the polarization energy and forces are computed using an interpolation between fully charges/polarizable and decharged/nonpolarizable ligand atoms as described previously.[162] Work is currently underway, in collaboration with Wei Yang (Chemistry, Florida State University) to implement the most recent versions of his orthogonal space tempering techniques into the family of Tinker programs.[163]

The setup of a Tinker keyfile for use of OSRW is straightforward. For instance, to compute the hydration of free energy of small solute in water only four additional keywords are required. First, the keyword LIGAND specifies the atom numbers of the solute for the hydration free energy calculation. The additional Tinker keywords OSRW-ABSOLUTE,

DONOLIGANDCONDENSED, DOVAPORELEC specify an absolute solvation energy calculation, the presence of only a single ligand molecule, and use of a gas phase leg in the free energy calculation, respectively.

Distance Geometry

In the context of molecular modeling, distance geometry (DG) is method for generating a structure or structures consistent with an input set of distance constraints. [164] [165] A basic DG algorithm takes an object in a high-dimensional mathematical "distance space", and reduces dimensionality by projecting it into a 3D molecular structure. An early important use of the method involved the generation of protein NMR structural models from short-range NMR NOE distance constraints. [166] However, a more interesting application of distance geometry is to under constrained problems. Given a limited set of upper and lower bound distances between atoms or groups in a molecular system, one would like for a distance geometry algorithm to generate a uniform sampling of all possible structures consistent with the input distance ranges. Tinker 8 contains an efficient method that exhibits excellent sampling properties for under constrained input through extension of standard DG algorithms. First, the Tinker *distgeom* program uses random partial metrization to update the matrix of upper and lower distance bounds whenever an individual distance value is fixed during structure generation. Only a small predetermined portion of the distance selections are followed by metrization, reducing the computational burden of a nominally $O(N^4)$ method.[167] Tinker uses a powerful, but relatively little-known, shortest path update algorithm to further reduce the metrization work load.[168] Second, *distgeom* selects distances between the upper and lower bounds from a Gaussian-like distribution tuned to reproduce reasonable molecule structures, instead of using the traditional flat, uniform distribution.[169] Additional terms are used to enforce local chirality and torsional constraints, and

simulated annealing on geometric constraints is used to refine output structures. The resulting Tinker program performs well in NMR applications,[170] and provided good sampling in less constrained situations such as protein structure prediction.[171]

6. Force Field Explorer

In addition to the suite of command line programs, Tinker also includes a graphical user interface (GUI) called Force Field Explorer or FFE. This program allows users to visualize molecular structures and provides access to many of Tinker's analysis, search and dynamics methods from a simple, user-friendly interface. This functionality makes FFE useful both as a research tool and as an instructional aid.

Force Field Explorer 8 gives users a powerful, simple and many-featured way to visualize molecular structures. It allows users to model molecules of interest using standard representations (wireframe, ball and stick, *etc.*). Molecules can be loaded directly from existing Tinker files, or downloaded from the NIH PubChem database,[172] the NCI CACTUS database and the RCSB Protein Data Bank (PDB).[173] Biopolymers can also be interactively constructed from sequence in various idealized structures. The program also gives users the ability to play back any Tinker molecular dynamics trajectory with the click of a button. In addition to these standard features, FFE also includes tools for force field-specific visualization. It can render a structure using the van der Waals radii specific to the force field being used, or display the partial charges or velocities assigned to each atom of a system. For polarizable force fields, it can display the induced dipoles as a vector at each atom at every time point of a simulation. These features allow users to assess in time and space how force field parameters affect the results of their calculations.

What makes Force Field Explorer a unique tool is that it combines visualization power with the functionality of Tinker. Through the graphical interface, users can run many of Tinker's analysis, search and dynamics programs. Simple minimizations or MD simulations can be started with the click of a button. The GUI has the ability to directly modify the Tinker key file via a graphical editing facility. By enabling access to the key file, users can quickly and easily change the options for whatever calculation they're running without touching the command line. As shown in the example of figure 6, FFE's functionality is laid out in an easy-to-navigate format. This combined with the integration with the full integration of Tinker makes Force Field Explorer useful not only for research, but also educational purposes.

Communication between FFE and Tinker is mediated by the Java sockets mechanism. Special versions of Tinker executables built against the FFE interface, allow Tinker calculations to send output to FFE in real time, including coordinates, velocities, induced dipoles, lattice parameters and other variables. Conversely, FFE is able to connect to an already running Tinker job on a remote machine, in order to perform job control tasks, display an MD trajectory interactively, *etc*.

7. Benchmarks

Six periodic boundary systems of increasing size (from 648 to 174219 atoms) have been constructed as benchmark tests to examine the efficacy of Tinker 8 and Tinker-OpenMM on standard CPU and commodity NVIDIA GPU devices, respectively. The systems reported include: a small water box of 216 AMOEBA water molecules, a larger 500 molecule TIP3P water box, the crystallographic unit cell of the plant protein crambin, a cucurbituril clip host-guest system from the SAMPL5 exercise,[174] a solvated DHFR protein, and a solvated COX-2 protein dimer. The system sizes differ by more than two orders of magnitude. Force fields tested

were Amber ff99SB[68] and AMOEBA. All simulations were performed with a 2 fs MD time step, and throughput is reported in nanoseconds per day in Table 2. We note that hydrogen mass reweighting,[175] which retards high-frequency motions, is a keyword option available in Tinker. Use of this option coupled with tight thermostating enables stable MD trajectories at 4 fs time steps, and yields roughly double the throughput reported in Table 2. As expected, the GPU implementation via Tinker-OpenMM significantly outperforms the reference CPU version of Tinker 8 for production MD calculations.

8. Conclusions & Future Development

As has been stressed throughout this report, a defining characteristic of the Tinker molecular mechanics package is its modularity. This intentional design lends itself to straightforward future development and software improvement. There are many unsolved problems requiring advanced energy models and sampling methods yet to be attacked by molecular modeling, and corresponding plans are underway for the future development of Tinker. There are three major projects currently in progress within the Tinker community: acceleration of the existing software, implementation of advanced potentials and sampling algorithms, and integration across the broader Tinker family of codes.

There are a host of problems in molecular biology and elsewhere where advanced models are needed but are computationally too inefficient to be tractable. Simulations of large RNA structures or proteins with significant conformational fluctuations have long been thought to be areas where advanced methods may be required. A future goal of the Tinker package is to make such simulations possible by improving the efficiency of advanced polarizable models. Techniques for speeding the costliest aspect of polarizable force fields, solution of the

polarization model itself, are under development for implementation in future versions of Tinker, as are support for current polarizable models including SIBFA[176] and GEM.[177]

In addition to efficient software for existing force fields, the Tinker project is developing code that will run the next generation of models. A new class of physics-based potentials is under development that relies less on empiricism than their predecessors. These models attempt to correct for errors that occur at short-range in point change and point multipole force fields because of overlapping charge distributions. Simple models to account for this effect on the electrostatic term of force fields, the so-called charge penetration error, have been recently published[178-180] and corresponding models for polarization, exchange-repulsion and dispersion are under development. These potentials are currently being incorporated into Tinker. We recognize that as computational power continues to grow, and the problems that molecular mechanics models are asked to solve become more demanding, it will be important to ensure that these new models have a home in Tinker.

Importantly, the future development of Tinker is directed toward unifying the code bases of the Tinker family of modeling packages, Tinker, Tinker-HP[181, 182] and Tinker-OpenMM. Because molecular mechanics simulations of large molecules remain computationally demanding, it is important that the full functionality of Tinker be available to users on a variety of hardware, from large scale CPU-based supercomputers to individual GPUs. The Tinker-HP and Tinker-OpenMM branches are responsible for enabling this high performance; Tinker-HP for massively parallel CPU calculations and Tinker-OpenMM as a CUDA-based GPU implementation. A goal of the Tinker project is to unify the code structure of each of these code packages. This has three major benefits. First, it will bring all of the codes up-to-date with the most efficient methods available. Second, future development of models or methods will be

more easily integrated across all three platforms if their structures are unified. Third, it will allow Open Source development of Tinker that can be propagated to the Tinker-HP and Tinker-OpenMM branches. By keeping Tinker-HP and Tinker-OpenMM in step with Tinker development, we can ensure users of access to Tinker functionality regardless of hardware platform.

The Tinker molecular modeling software package is an easy-to-use, easy-to-understand and easy-to-modify set of programs allowing researchers to model molecular systems of interest in a variety of ways. It supports a broad spectrum of classical molecular mechanics models as well as an array of algorithms to efficiently explore the corresponding potential energy surfaces. This is accomplished through a modular code structure that permits users to inspect and manipulate calculation details, and developers to add new functionality quickly. Because it is Open Source and freely available to academics, Tinker 8 provides a community code base in which to test old ideas and investigate new ones. It is our hope that this community-oriented model will continue to advance development of tools that make the Tinker toolbox useful.

Acknowledgements

JWP and PR wish to thank the National Institutes of Health NIGMS for support of recent force field and software development via awards R01 GM106137 and R01 GM114237. Tinker, as with most large software packages under development for many years, has a very large number of contributors– far too many to list here– who have provided code and suggestions. JWP, in particular, is grateful for help from a wide community of colleagues, developers and users stretching over more than three decades.

References

1. Lagardère L, Jolly L-H, Lipparini F, Aviat F, Stamm B, Jing ZF, Harger M, Torabifard H, Cisneros GA, Schnieders MJ, Gresh N, Maday Y, Ren PY, Ponder JW, Piquemal J-P. Tinker-HP: A Massively Parallel Molecular Dynamics Package for Multiscale Simulations of Large Complex Systems with Advanced Point Dipole Polarizable Force Fields. ***Chem. Sci.,*** *8,* 956-72 (2018).
2. Harger M, Li D, Wang Z, Dalby K, Larardere L, Piquemal J-P, Ponder JW, Ren PY. Tinker-OpenMM: Absolute and Relative Alchemical Free Energies Using AMOEBA on GPUs. ***J. Comput. Chem.,*** *38,* 2047-55 (2017).
3. Ponder JW. Tinker Molecular Modeling. Washington University in St. Louis; 2018 Available from: https://dasher.wustl.edu/tinker/.
4. Piquemal J-P. Piquemal Research & Software. Sorbonne Universites; 2018 Available from: http://piquemalresearch.com/research-and-softwares/.
5. Ren P. Tinker GPU Main Page. University of Texas, Austin; 2018 Available from: http://biomol.bme.utexas.edu/tinkergpu/.
6. Ponder JW. Tinker: Software Tools for Molecular Design. GitHub; 2018 Available from: https://github.com/jayponder/tinker/.
7. Ren P. Tinker-OpenMM Toolkit for Molecular Simulation Using High Performance GPU Code. GitHub; 2018 Available from: https://github.com/pren/tinker-openmm/.
8. Allinger NL. Conformational Analysis. 130. MM2. A Hydrocarbon Force Field Utilizing V1 and V2 Torsional Terms. ***J. Am. Chem. Soc.,*** *99,* 8127-34 (1977).
9. Allinger NL, Yuh YH, Lii JH. Molecular Mechanics. The MM3 Force Field for Hydrocarbons. 1. ***J. Am. Chem. Soc.,*** *111,* 8551-66 (1989).
10. Corey EJ, Ponder JW. Stereochemistry of the Hygrolidins. ***Tetrahedron Lett.,*** *25,* 4325-8 (1984).
11. ChemOffice. CambridgeSoft.com; 2018.
12. Ponder JW, Richards FM. An Efficient Newton‐like Method for Molecular Mechanics Energy Minimization of Large Molecules. ***J. Comput. Chem.,*** *8,* 1016-24 (1987).
13. Ponder JW, Richards FM. Tertiary Templates for Proteins: Use of Packing Criteria in the Enumeration of Allowed Sequences for Different Structural Classes. ***J. Mol. Biol.,*** *193,* 775-91 (1987).
14. Pande VS, Baker I, Chapman J, Elmer SP, Khaliq S, Larson SM, Rhee YM, Shirts MR, Snow CD, Sorin EJ, Zagrovic B. Atomistic Protein Folding Simulations on the Submillisecond Time Scale Using Worldwide Distributed Computing. ***Biopolymers,*** *68,* 91-109 (2003).
15. Humphrey W, Dalke A, Schulten K. VMD - Visual Molecular Dynamics. ***J. Mol. Graphics,*** *14,* 33-8 (1996).
16. DeLano WL. PyMOL: An Open-Source Molecular Graphics Tool. ***CCP4 Newsletter on Protein Crystallography,*** *40,* 82-92 (2002).
17. Hanson RM. Jmol - A Paradigm Shift in Crystallographic Visualization. ***J. Appl. Crystallogr.,*** *43,* 1250-60 (2010).
18. LuCore SD, Litman JM, Powers KT, Gao S, Lynn AM, Tollefson WTA, Fenn TD, Washington MT, Schnieders MJ. Dead-End Elimination with a Polarizable Force Field Repacks PCNA Structures. ***Biophys. J,*** *109,* 816-26 (2015).

19. O'Boyle NM, Banck M, James CA, Morley C, Vendermeersch T, Hutchison GR. Open Babel: An Open Chemical Toolbox. ***J. Cheminformatics,*** *3,* 33 (2011).
20. McGibbon RT, Beauchamp KA, Harrigan MP, Klein C, Swails J, Hernandez CX, Schwantes CR, Wang L-P, Lane TJ, Pande VS. MDTraj: A Modern Open Library for the Analysis of Molecular Dynamics Trajectories. ***Biophys. J,*** *109,* 1528-32 (2015).
21. Michaud-Agrawal M, Denning EJ, Woolf TB, Beckstein O. MDAnalysis: A Toolkit for the Analysis of Molecular Dynamics Simulations. ***J. Comput. Chem.,*** *32,* 2319-27 (2011).
22. Swails J, Hernandez C, Mobley D, Nguyen H, Wang L-P, Janowski P. ParmEd: Parameter/Topology Editor and Molecular Simulator. 2018 Available from: https://github.com/ParmEd/ParmEd.
23. Schaftenaar G, Vlieg E, Vriend G. Molden 2.0: Quantum Chemistry Meets Proteins. ***J Comput. Aid. Mol. Des.,*** *31,* 789-800 (2017).
24. Pedretti A, Villa L, Vistoli G. VEGA - An Open Platform to Develop Chemo-Bio-Informatics Applications, Using Plug-In Architecture and Script Programming. ***J Comput. Aid. Mol. Des.,*** *18,* 167-73 (2004).
25. Martinez L, Andrade R, Birgin EG, Martinez JM. Packmol: A Package for Building Initiial Configurations for Molecular Dynamics Simulations. ***J. Comput. Chem.,*** *30,* 2157-64 (2009).
26. Wang L-P, Martinez TJ, Pande VS. Building Force Fields: An Automatic, Systematic, and Reproducible Approach. ***J. Phys. Chem. Lett.,*** *5,* 1885-91 (2014).
27. Schmidt JR, Polik WF. WebMO Enterprise, Version 13.0. Holland, MI: WebMO LLC; 2013.
28. Woodcock HL, Miller BT, Hodoscek M, Okru A, Larkin JD, Ponder JW, Brooks BR. MSCALE: A General Utility for Multiscale Modeling. ***J. Chem. Theory Comput.,*** *7,* 1208-19 (2011).
29. Eastman P, Friedrichs MS, Chodera JD, Radmer RJ, Bruns CM, Ku JP, Beauchamp KA, Lane TJ, Wang L-P, Shukla D. OpenMM 4: A Reusable, Extensible, Hardware Independent Library for High Performance Molecular Simulation. ***J. Chem. Theory Comput.,*** *9,* 461-9 (2012).
30. Eastman P, Swails J, Chodera JD, McGibbon RT, Zhao Y, Beauchamp KA, Wang L-P, Simmonett AC, Harrigan MP, Stern CD, Wiewiora RP, Brooks BR, Pande VS. OpenMM 7: Rapid Development of High Performance Algorithms for Molecular Dynamics. ***PLoS Comput. Biol.,*** *13,* e1005659 (2017).
31. Huang J, Rauscher S, Nawrocki G, Ran T, Feig M, de Groot BL, Grubmüller H, MacKerell Jr AD. CHARMM36m: An Improved Force Field for Folded and Intrinsically Disordered Proteins. ***Nat. Methods,*** *14,* 71-3 (2016).
32. Maier JA, Martinez C, Kasavajhala K, Wickstrom L, Hauser KE, Simmerling C. ff14SB: Improving the Accuracy of Protein Side Chain and Backbone Parameters from ff99SB. ***J. Chem. Theory Comput.,*** *11,* 3696-713 (2015).
33. Robertson MJ, Tirado-Rives J, Jorgensen WL. Improved Peptide and Protein Torsional energetics with the OPLS-AA Force Field. ***J. Chem. Theory Comput.,*** *11,* 3499-509 (2015).
34. Palmo K, Mannfors B, Mirkin NG, Krimm S. Potential Energy Functions: From Consistent Force Fields to Spectroscopically Determined Polarizable Force Fields. ***Biopolymers,*** *68,* 383-94 (2003).

35. Wilson Jr EB, Decius JG, Cross PG, Lagemann RT. Molecular Vibrations. ***Am. J. Phys.,*** *23,* 550- (1955).
36. Liljefors T, Tai JC, Li S, Allinger NL. On the Out‐of‐Plane Deformation of Aromatic Rings, and Its Representation by Molecular Mechanics. ***J. Comput. Chem.,*** *8,* 1051-6 (1987).
37. Urey HC, Bradley Jr CA. The Vibrations of Pentatonic Tetrahedral Molecules. ***Phys. Rev.,*** *38,* 1969-78 (1931).
38. Lennard-Jones JE. Cohesion. ***P. Phys. Soc.,*** *43,* 461-82 (1931).
39. Halgren TA. Representation of van der Waals (vdW) Interactions in Molecular Mechanics Force Fields: Potential Form, Combination Rules, and vdW Parameters. ***J. Am. Chem. Soc.,*** *114,* 7827-43 (1992).
40. Buckingham RA. The Classical Equation of State of Gaseous Helium, Neon and Argon. ***P. Roy. Soc. Lond. A Mat.,*** *168,* 264-83 (1938).
41. Lii JH, Allinger NL. Directional Hydrogen Bonding in the MM3 Force Field. I. ***J. Phys. Org. Chem.,*** *7,* 591-609 (1994).
42. Lii JH, Allinger NL. Directional Hydrogen Bonding in the MM3 Force Field: II. ***J. Comput. Chem.,*** *19,* 1001-16 (1998).
43. Ponder JW, Wu C, Ren P, Pande VS, Chodera JD, Schnieders MJ, Haque I, Mobley DL, Lambrecht DS, DiStasio Jr RA. Current Status of the AMOEBA Polarizable Force Field. ***J. Phys. Chem. B,*** *114,* 2549-64 (2010).
44. Ren P, Ponder JW. Polarizable Atomic Multipole Water Model for Molecular Mechanics Simulation. ***J. Phys. Chem. B,*** *107,* 5933-47 (2003).
45. Ren P, Wu C, Ponder JW. Polarizable Atomic Multipole-based Molecular Mechanics for Organic Molecules. ***J. Chem. Theory Comput.,*** *7,* 3143-61 (2011).
46. Shi Y, Xia Z, Zhang J, Best R, Wu C, Ponder JW, Ren P. Polarizable Atomic Multipole-based AMOEBA Force Field for Proteins. ***J. Chem. Theory Comput.,*** *9,* 4046-63 (2013).
47. Zhang C, Lu C, Jing Z, Wu C, Piquemal J-P, Ponder JW, Ren P. AMOEBA Polarizable Atomic Multipole Force Field for Nucleic Acids. ***J. Chem. Theory Comput.,*** *14,* 2084-108 (2018).
48. Onufriev A, Case DA, Bashford D. Effective Born Radii in the Generalized Born Approximation: The Importance of Being Perfect. ***J. Comput. Chem.,*** *23,* 1297-304 (2002).
49. Schaefer M, Bartels C, Leclerc F, Karplus M. Effective Atom Volumes for Implicit Solvent Models: Comparison between Voronoi Volumes and Minimum Fluctuations Volumes. ***J. Comput. Chem.,*** *22,* 1857-79 (2001).
50. Grycuk T. Deficiency of the Coulomb-Field Approximation in the Generalized Born Model: An Improved Formula for Born Radii Evaluation. ***J. Chem. Phys.,*** *119,* 4817-26 (2003).
51. Schnieders MJ, Ponder JW. Polarizable Atomic Multipole Solutes in a Generalized Kirkwood Continuum. ***J. Chem. Theory Comput.,*** *3,* 2083-97 (2007).
52. Wesson L, Eisenberg D. Atomic Solvation Parameters Applied to Molecular Dynamics of Proteins in Solution. ***Protein Sci.,*** *1,* 227-35 (1992).
53. Lin MS, Fawzi NL, Head-Gordon T. Hydrophobic Potential of Mean Force as a Solvation Function for Protein Structure Prediction. ***Structure,*** *15,* 727-40 (2007).

54. Kong Y, Ponder JW. Calculation of the Reaction Field due to Off-Center Point Multipoles. ***J. Chem. Phys.,*** *107,* 481-92 (1997).
55. Warwicker J, Watson HC. Calculation of the Electric Potential in the Active Site Cleft due to α-Helix Dipoles. ***J. Mol. Biol.,*** *157,* 671-9 (1982).
56. Klapper I, Hagstrom R, Fine R, Sharp K, Honig B. Focusing of Electric Fields in the Active Site of Cu ‐ Zn Superoxide Dismutase: Effects of Ionic Strength and Amino ‐ Acid Modification. ***Proteins,*** *1,* 47-59 (1986).
57. Sharp KA, Honig B. Electrostatic Interactions in Macromolecules: Theory and Applications. ***Annu. Rev. Biophys. Bio.,*** *19,* 301-32 (1990).
58. Baker NA, Sept D, Joseph S, Holst MJ, McCammon JA. Electrostatics of Nanosystems: Application to Microtubules and the Ribosome. ***P. Natl. Acad. Sci. USA.,*** *98,* 10037-41 (2001).
59. Schnieders MJ, Baker NA, Ren P, Ponder JW. Polarizable Atomic Multipole Solutes in a Poisson-Boltzmann Continuum. ***J. Chem. Phys.,*** *126,* 124114 (2007).
60. Allinger NL, Li F, Yan L, Tai JC. Molecular Mechanics (MM3) Calculations on Conjugated Hydrocarbons. ***J. Comput. Chem.,*** *11,* 868-95 (1990).
61. Xiang JY, Ponder JW. A Valence Bond Model for Aqueous Cu(II) and Zn(II) Ions in the AMOEBA Polarizable Force Field. ***J. Comput. Chem.,*** *34,* 739-49 (2013).
62. Xiang JY, Ponder JW. An Angular Overlap Model for Cu(II) Ion in the AMOEBA Polarizable Force Field. ***J. Chem. Theory Comput.,*** *10,* 298-311 (2014).
63. Carlsson AE, Zapata S. The Functional Form of Angular Forces around Transition Metal Ions in Biomolecules. ***Biophys. J,*** *81,* 1-10 (2001).
64. Cornell WD, Cieplak P, Bayly CI, Gould IR, Merz KM, Ferguson DM, Spellmeyer DC, Fox T, Caldwell JW, Kollman PA. A Second Generation Force Field for the Simulation of Proteins, Nucleic Acids, and Organic Molecules. ***J. Am. Chem. Soc.,*** *117,* 5179-97 (1995).
65. Kollman P, Dixon R, Cornell W, Fox T, Chipot C, Pohorille A. The Development/Application of a 'Minimalist' Organic/Biochemical Molecular Mechanic Force Field Using a Combination of ab Initio Calculations and Experimental Data. Computer Simulation of Biomolecular Systems, Vol. 3: Springer, Dordrecht; 1997. p. 83-96.
66. Cheatham III TE, Cieplak P, Kollman PA. A Modified Version of the Cornell et al. Force Field with Improved Sugar Pucker Phases and Helical Repeat. ***J. Biomol. Struct. Dyn.,*** *16,* 845-62 (1999).
67. Wang J, Cieplak P, Kollman PA. How Well Does a Restrained Electrostatic Potential (RESP) Model Perform in Calculating Conformational Energies of Organic and Biological Molecules? ***J. Comput. Chem.,*** *21,* 1049-74 (2000).
68. Hornak V, Abel R, Okur A, Strockbine B, Roitberg A, Simmerling C. Comparison of Multiple Amber Force Fields and Development of Improved Protein Backbone Parameters. ***Proteins,*** *65,* 712-25 (2006).
69. Neria E, Fischer S, Karplus M. Simulation of Activation Free Energies in Molecular Systems. ***J. Chem. Phys.,*** *105,* 1902-21 (1996).
70. MacKerell Jr AD, Bashford D, Bellott MLDR, Dunbrack Jr RL, Evanseck JD, Field MJ, Fischer S, Gao J, Guo H, Ha S. All-Atom Empirical Potential for Molecular Modeling and Dynamics Studies of Proteins. ***J. Phys. Chem. B,*** *102,* 3586-616 (1998).

71. Foloppe N, MacKerell Jr AD. All‐Atom Empirical Force Field for Nucleic Acids: I. Parameter Optimization Based on Small Molecule and Condensed Phase Macromolecular Target Data. ***J. Comput. Chem.,*** *21,* 86-104 (2000).
72. MacKerell Jr AD, Feig M, Brooks III CL. Extending the Treatment of Backbone Energetics in Protein Force Fields: Limitations of Gas‐Phase Quantum Mechanics in Reproducing Protein Conformational Distributions in Molecular Dynamics Simulations. ***J. Comput. Chem.,*** *25,* 1400-15 (2004).
73. Jorgensen WL, Maxwell DS, Tirado-Rives J. Development and Testing of the OPLS All-Atom Force Field on Conformational Energetics and Properties of Organic Liquids. ***J. Am. Chem. Soc.,*** *118,* 11225-36 (1996).
74. Kaminski GA, Friesner RA, Tirado-Rives J, Jorgensen WL. Evaluation and Reparametrization of the OPLS-AA Force Field for Proteins via Comparison with Accurate Quantum Chemical Calculations on Peptides. ***J. Phys. Chem. B,*** *105,* 6474-87 (2001).
75. Weiner SJ, Kollman PA, Case DA, Singh UC, Ghio C, Alagona G, Profeta S, Weiner P. A New Force Field for Molecular Mechanical Simulation of Nucleic Acids and Proteins. ***J. Am. Chem. Soc.,*** *106,* 765-84 (1984).
76. Jorgensen WL, Severance DL. Aromatic-Aromatic Interactions: Free Energy Profiles for the Benzene Dimer in Water, Chloroform, and Liquid Benzene. ***J. Am. Chem. Soc.,*** *112,* 4768-74 (1990).
77. Maxwell DS, Tirado‐Rives J, Jorgensen WL. A Comprehensive Study of the Rotational Energy Profiles of Organic Systems by ab Initio MO Theory, Forming a Basis for Peptide Torsional Parameters. ***J. Comput. Chem.,*** *16,* 984-1010 (1995).
78. Jorgensen WL, Tirado-Rives J. The OPLS (Optimized Potentials for Liquid Simulations) Potential Functions for Proteins, Energy Minimizations for Crystals of Cyclic Peptides and Crambin. ***J. Am. Chem. Soc.,*** *110,* 1657-66 (1988).
79. Sprague JT, Tai JC, Yuh YH, Allinger NL. The MMP2 Calculational Method. ***J. Comput. Chem.,*** *8,* 581-603 (1987).
80. Allinger NL, Kok RA, Imam MR. Hydrogen Bonding in MM2. ***J. Comput. Chem.,*** *9,* 591-5 (1988).
81. Lii JH, Allinger NL. 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,* 8576-82 (1989).
82. Allinger NL, Li F, Yan L. Molecular Mechanics. The MM3 Force Field for Alkenes. ***J. Comput. Chem.,*** *11,* 848-67 (1990).
83. Lii JH, Allinger NL. The MM3 Force Field for Amides, Polypeptides and Proteins. ***J. Comput. Chem.,*** *12,* 186-99 (1991).
84. Halgren TA, Nachbar RB. Merck Molecular Force Field. IV. Conformational Energies and Geometries for MMFF94. ***J. Comput. Chem.,*** *17,* 587-615 (1996).
85. Ren P, Ponder JW. Consistent Treatment of Inter‐and Intramolecular Polarization in Molecular Mechanics Calculations. ***J. Comput. Chem.,*** *23,* 1497-506 (2002).
86. Wu JC, Piquemal J-P, Chaudret R, Reinhardt P, Ren P. Polarizable Molecular Dynamics Simulation of Zn (II) in Water Using the AMOEBA Force Field. ***J. Chem. Theory Comput.,*** *6,* 2059-70 (2010).

87. Grossfield A, Ren P, Ponder JW. Ion Solvation Thermodynamics from Simulation with a Polarizable Force Field. ***J. Am. Chem. Soc.,*** *125,* 15671-82 (2003).
88. Dang LX. Development of Nonadditive Intermolecular Potentials Using Molecular Dynamics: Solvation of Li+ and F− Ions in Polarizable Water. ***J. Chem. Phys.,*** *96,* 6970-7 (1992).
89. Smith DE, Dang LX. Interionic Potentials of Mean Force for SrCl2 in Polarizable Water: A Computer Simulation Study. ***Chem. Phys. Lett.,*** *230,* 209-14 (1994).
90. Dang LX, Chang T-M. Molecular Dynamics Study of Water Clusters, Liquid, and Liquid–Vapor Interface of Water with Many-Body Potentials. ***J. Chem. Phys.,*** *106,* 8149-59 (1997).
91. Chang T-M, Dang LX. Detailed Study of Potassium Solvation Using Molecular Dynamics Techniques. ***J. Phys. Chem. B,*** *103,* 4714-20 (1999).
92. Dang LX. Intermolecular Interactions of Liquid Dichloromethane and Equilibrium Properties of Liquid–Vapor and Liquid–Liquid Interfaces: A Molecular Dynamics Study. ***J. Chem. Phys.,*** *110,* 10113-22 (1999).
93. Chang T-M, Dang LX. On Rotational Dynamics of an NH4+ Ion in Water. ***J. Chem. Phys.,*** *118,* 8813-20 (2003).
94. Dang LX, Schenter GK, Glezakou V-A, Fulton JL. Molecular Simulation Analysis and X-Ray Absorption Measurement of Ca2+, K+ and Cl-Ions in Solution. ***J. Phys. Chem. B,*** *110,* 23644-54 (2006).
95. Wick CD, Dang LX. Molecular Dynamics Study of Ion Transfer and Distribution at the Interface of Water and 1, 2-Dichlorethane. ***J. Phys. Chem. C,*** *112,* 647-9 (2008).
96. Sun X, Chang T-M, Cao Y, Niwayama S, Hase WL, Dang LX. Solvation of Dimethyl Succinate in a Sodium Hydroxide Aqueous Solution. A Computational Study. ***J. Phys. Chem. B,*** *113,* 6473-7 (2009).
97. Dang LX, Truong TB, Ginovska-Pangovska B. Note: Interionic Potentials of Mean Force for Ca2+-Cl− in Polarizable Water. ***J. Chem. Phys.,*** *136,* 126101 (2012).
98. Kearsley SK. On the Orthogonal Transformation Used for Structural Comparisons. ***Acta Crystallogr. A,*** *45,* 208-10 (1989).
99. Liu DC, Nocedal J. On the Limited Memory BFGS Method for Large Scale Optimization. ***Math. Program.,*** *45,* 503-28 (1989).
100. Nocedal J. Updating Quasi-Newton Matrices with Limited Storage. ***Math. Comput.,*** *35,* 773-82 (1980).
101. Wright S, Nocedal J. Numerical Optimization, 2nd Ed. New York, NY: Springer Science; 1999.
102. Shanno DF, Phua K-H. Numerical Comparison of Several Variable-Metric Algorithms. ***J. Optimiz. Theory App.,*** *25,* 507-18 (1978).
103. Davidon WC. Optimally Conditioned Optimization Algorithms without Line Searches. ***Math. Program.,*** *9,* 1-30 (1975).
104. Dembo RS, Steihaug T. Truncated-Newton Algorithms for Large-Scale Unconstrained Optimization. ***Math. Program.,*** *26,* 190-212 (1983).
105. Halgren TA, Lipscomb WN. The Synchronous-Transit Method for Determining Reaction Pathways and Locating Molecular Transition States. ***Chem. Phys. Lett.,*** *49,* 225-32 (1977).

106. Behn A, Zimmerman PM, Head-Gordon M. Incorporating Linear Synchronous Transit Interpolation into the Growing String Method: Algorithm and Applications. ***J. Chem. Theory Comput.,*** *7,* 4019-25 (2011).
107. Bell S, Crighton JS. Locating Transition States. ***J. Chem. Phys.,*** *80,* 2464-75 (1984).
108. Czerminski R, Elber R. Reaction Path Study of Conformational Transitions in Flexible Systems: Applications to Peptides. ***J. Chem. Phys.,*** *92,* 5580-601 (1990).
109. Moré JJ, Garbow BS, Hillstrom KE. User Guide for MINPACK-1, Argonne National Laboratory Report ANL-80-74. Argonne, IL; 1980Contract.
110. Kirkpatrick S, Gelatt CD, Vecchi MP. Optimization by Simulated Annealing. ***Science,*** *220,* 671-80 (1983).
111. Griewank AO. Generalized Descent for Global Optimization. ***J. Optimiz. Theory App.,*** *34,* 11-39 (1981).
112. Butler RAR, Slaminka EE. An Evaluation of the Sniffer Global Optimization Algorithm Using Standard Test Functions. ***J. Comput. Phys.,*** *99,* 28-32 (1992).
113. Kolossváry I, Guida WC. Low‐Mode Conformational Search Elucidated: Application to C39H80 and Flexible Docking of 9‐Deazaguanine Inhibitors into PNP. ***J. Comput. Chem.,*** *20,* 1671-84 (1999).
114. Li Z, Scheraga HA. Monte Carlo-Minimization Approach to the Multiple-Minima Problem in Protein Folding. ***P. Natl. Acad. Sci. USA.,*** *84,* 6611-5 (1987).
115. Wales DJ, Doye JPK. Global Optimization by Basin-Hopping and the Lowest Energy Structures of Lennard-Jones Clusters Containing up to 110 Atoms. ***Journal of Physical Chemistry A,*** *101,* 5111-6 (1997).
116. Kostrowicki J, Scheraga HA. Application of the Diffusion Equation Method for Global Optimization to Oligopeptides. ***J. Phys. Chem.,*** *96,* 7442-9 (1992).
117. Nakamura S, Hirose H, Ikeguchi M, Doi J. Conformational Energy Minimization Using a Two-Stage Method. ***J. Phys. Chem.,*** *99,* 8374-8 (1995).
118. Pappu RV, Hart RK, Ponder JW. Analysis and Application of Potential Energy Smoothing for Global Optimization. ***J. Phys. Chem. B,*** *102,* 9725-42 (1998).
119. Pappu RV, Marshall GR, Ponder JW. A Potential Smoothing Algorithm Accurately Predicts Transmembrane Helix Packing. ***Nat. Struct. Biol.,*** *6,* 50-5 (1999).
120. Ma J, Straub JE. Simulated Annealing Using the Classical Density Distribution. ***J. Chem. Phys.,*** *101,* 533-41 (1994).
121. Rogers J, J. W., Donnelly RA. Potential Transformation Methods for Large-Scale Global Optimization. ***SIAM J. Optimiz.,*** *5,* 871-91 (1995).
122. Beeman D. Some Multistep Methods for Use in Molecular Dynamics Calculations. ***J. Comput. Phys.,*** *20,* 130-9 (1976).
123. Brooks BR. Algorithms for Molecular Dynamics at Constant Temperature and Pressure. ***DCRT Report, NIH,*** (1988).
124. Lelièvre T, Rousset M, Stoltz G. Langevin Dynamics with Constraints and Computation of Free Energy Differences. ***Math. Comput.,*** *81,* 2071-125 (2012).
125. Lelièvre T, Stoltz G, Rousset M. Free Energy Computations: A Mathematical Perspective. London, UK: Imperial College Press; 2010.
126. Martyna GJ, Tuckerman ME, Tobias DJ, Klein ML. Explicit Reversible Integrators for Extended Systems Dynamics. ***Mol. Phys.,*** *87,* 1117-57 (1996).

127. Bussi G, Zykova-Timan T, Parrinello M. Isothermal-Isobaric Molecular Dynamics Using Stochastic Velocity Rescaling. ***J. Chem. Phys.,*** *130,* 074101 (2009).
128. Qian X, Schlick T. Efficient Multiple-Time-Step Integrators with Distance-based Force Splitting for Particle-Mesh-Ewald Molecular Dynamics Simulations. ***J. Chem. Phys.,*** *116,* 5971-83 (2002).
129. Humphreys DD, Friesner RA, Berne BJ. A Multiple-Time-Step Molecular Dynamics Algorithm for Macromolecules. ***J. Phys. Chem.,*** *98,* 6885-92 (1994).
130. Smith W. Hail Euler and Farewell: Rotational Motion in the Laboratory Frame. ***CCP5 Newsletter, Feb.,*** (2005).
131. Andersen HC. Rattle: A "Velocity" Version of the Shake Algorithm for Molecular Dynamics Calculations. ***J. Comput. Phys.,*** *52,* 24-34 (1983).
132. Allen MP. Brownian Dynamics Simulation of a Chemical Reaction in Solution. ***Mol. Phys.,*** *40,* 1073-87 (1980).
133. Guarnieri F, Still WC. A Rapidly Convergent Simulation Method: Mixed Monte Carlo/Stochastic Dynamics. ***J. Comput. Chem.,*** *15,* 1302-10 (1994).
134. Yun-Yi S, Lu W, van Gunsteren WF. On the Approximation of Solvent Effects on the Conformation and Dynaics of Cyclosporin A by Stochastic Dynamics Simulation Techniques. ***Mol. Simulat.,*** *1,* 369-83 (1988).
135. Bussi G, Donadio D, Parrinello M. Canonical Sampling through Velocity Rescaling. ***J. Chem. Phys.,*** *126,* 014101 (2007).
136. Berendsen HJC, van Postma JPM, van Gunsteren WF, DiNola ARHJ, Haak JR. Molecular Dynamics with Coupling to an External Bath. ***J. Chem. Phys.,*** *81,* 3684-90 (1984).
137. Andersen HC. Molecular Dynamics Simulations at Constant Pressure and/or Temperature. ***J. Chem. Phys.,*** *72,* 2384-93 (1980).
138. Evans DJ, Holian BL. The Nose–Hoover Thermostat. ***J. Chem. Phys.,*** *83,* 4069-74 (1985).
139. Frenkel D, Smit B. Understanding Molecular Simulation: From Algorithms to Applications, 2nd Ed. New York, NY: Academic Press; 2001.
140. Leimkuhler B, Margul DT, Tuckerman ME. Stochastic, Resonance-Free Multiple Time-Step Algorithm for Molecular Dynamics with Very Large Time Steps. ***Mol. Phys.,*** *111,* 3579-94 (2013).
141. Minary P, Martyna GJ, Tuckerman ME. Algorithms and Novel Applications based on the Isokinetic Ensemble. I. Biophysical and Path Integral Molecular Dynamics. ***J. Chem. Phys.,*** *118,* 2510-26 (2003).
142. Kaledin AL, Kaledin M, Bowman JM. All-Atom Calculation of the Normal Modes of Bacteriorhodopsin Using a Sliding Block Iterative Diagonalization Method. ***J. Chem. Theory Comput.,*** *2,* 166-74 (2006).
143. Zwanzig RW. High‐Temperature Equation of State by a Perturbation Method. I. Nonpolar Gases. ***J. Chem. Phys.,*** *22,* 1420-6 (1954).
144. Bennett CH. Efficient Estimation of Free Energy Differences from Monte Carlo Data. ***J. Comput. Phys.,*** *22,* 245-68 (1976).
145. Daly KB, Benziger JB, Debenedetti PG, Panagiotopoulos AZ. Massively Parallel Chemical Potential Calculation on Graphics Processing Units. ***Comput. Phys. Commun.,*** *183,* 2054-62 (2012).

146. Wyczalkowski MA, Vitalis A, Pappu RV. New Estimators for Calculating Solvation Entropy and Enthalpy and Comparative Assessments of their Accuracy and Precision. ***J. Phys. Chem. B,*** *114,* 8166-80 (2010).
147. Bell DR, Qi R, Jing Z, Xiang JY, Meijas C, Schnieders MJ, Ponder JW, Ren P. Calculating Binding Free Energies fo Host-Guest Systems Using the AMOEBA Polarizable Force Field. ***Phys. Chem. Chem. Phys.,*** *18,* 30261-9 (2016).
148. Stone AJ. Distributed Multipole Analysis: Stability for Large Basis Sets. ***J. Chem. Theory Comput.,*** *1,* 1128-32 (2005).
149. Wu JC, Chattree G, Ren P. Automation of AMOEBA Polarizable Force Field Parameterization for Small Molecules. ***Theor. Chem. Acc.,*** *131,* 1138 (2012).
150. Sullivan F, Mountain RD, O'Connellf J. Molecular Dynamics on Vector Computers. ***J. Comput. Phys.,*** *61,* 138-53 (1985).
151. Sagui C, Pedersen LG, Darden TA. Towards an Accurate Representation of electrostatics in Classical Force Fields: Efficient Implementation of Multipolar Interactions in Biomolecular Simulations. ***J. Chem. Phys.,*** *120,* (2004).
152. Frigo M, Johnson SG. The Design and Implementation of FFTW3. ***P. IEEE,*** *93,* 216-31 (2005).
153. Wang W, Skeel RD. Fast Evaluation of Polarizable Forces. ***J. Chem. Phys.,*** *123,* 164107 (2005).
154. Simmonett AC, Pickard IV FC, Shao Y, Cheatham III TE, Brooks BR. Efficient Treatment of Induced Dipoles. ***J. Chem. Phys.,*** *143,* 074115 (2015).
155. Simmonett AC, Pickard IV FC, Ponder JW, Brooks BR. An Empirical Extrapolation Scheme for Efficient Treatment of Induced Dipoles. ***J. Chem. Phys.,*** *145,* 164101 (2016).
156. Albaugh A, Demerdash O, Head-Gordon T. An Efficient and Stable Hybrid Extended Lagrangian/Self-Consistent Field Scheme for Solving Classical Mutual Induction. ***J. Chem. Phys.,*** *143,* 174104 (2015).
157. Albaugh A, Niklasson AMN, Head-Gordon T. Accurate Classical Polarization Solution with No Self-Consistent Field Iterations. ***J. Phys. Chem. Lett.,*** *8,* 1714-23 (2017).
158. Aviat F, Lagardère L, Piquemal J-P. The Truncated Conjugate Gradient (TCG), a Non-Iterative/Fixed-Cost Strategy for Computing Polarization in Molecular Dynamics: Fast Evaluation of Analytical Forces. ***J. Chem. Phys.,*** *147,* 161724 (2017).
159. Zheng L, Chen M, Yang W. Random Walk in Orthogonal Space to Achieve Efficient Free-Energy Simulation of Complex Systems. ***P. Natl. Acad. Sci. USA.,*** *105,* 20227-32 (2008).
160. Zheng L, Chen M, Yang W. Simultaneous Escaping of Explicit and Hidden Free Energy Barriers: Application of the Orthogonal Space Random Walk Strategy in Generalized Ensemble Based Conformational Sampling. ***J. Chem. Phys.,*** *130,* 06B618 (2009).
161. Abella JR, Cheng SY, Wang Q, Yang W, Ren P. Hydration Free Energy from Orthogonal Space Random Walk and Polarizable Force Field. ***J. Chem. Theory Comput.,*** *10,* 2792-801 (2014).
162. Schnieders MJ, Baltrusaitis J, Shi Y, Chattree G, Zheng L, Yang W, Ren P. The Structure, Thermodynamics, and Solubility of Organic Crystals from Simulation with a Polarizable Force Field. ***Journal of Chemical Theory aand Computation,*** *8,* 1721-36 (2012).

163. Zheng L, Yang W. Practically Efficient and Robust Free Energy Calculations: Double-Integration Orthogonal Space Tempering. ***J. Chem. Theory Comput.,*** *8,* 810-23 (2012).
164. Crippen GM, Havel TF. Distance Geometry and Molecular Conformation. Somerset, England: Research Studies Press, Ltd.; 1988.
165. Mucherino A, Lavor C, Liberti L, Maculan N. Distance Geometry: Theory, Methods and Applications. New York: Springer; 2013.
166. Wuthrich K. NMR of Proteins and Nucleic Acids. New York: Wiley-Interscience; 1986.
167. Kuszewski J, Nilges M, Brunger AT. Sampling and Efficiency of Metric Matrix Distance Geometry: A Novel Partial Metrization Algorithm. ***J. Biomol. NMR,*** *2,* 33-56 (1992).
168. Dionne R. Étude et Extension d'un Algorithme de Murchland. ***INFOR,*** *16,* 132-46 (1978).
169. Oshiro CM, Thomason J, Kuntz ID. Effects of Limited Input Distance Constraints Upon the Distance Geometry Algorithm. ***Biopolymers,*** *31,* 1049-64 (1991).
170. Hodsdon ME, Ponder JW, Cistola DP. The NMR Solution Structure of Intestinal Fatty Acid-Binding Protein Complexed with Palmitate: Application of a Novel Distance Geometry Algorithm. ***J. Mol. Biol.,*** *264,* 585-602 (1996).
171. Huang ES, Samudrala R, Ponder JW. Distance Geometry Generates Native-like Folds for Small Helical Proteins Using Consensus Distances of Predicted Protein Structures. ***Protein Sci.,*** *7,* 1998-2003 (1998).
172. Kim S, Thiessen PA, Bolton EE, Chen J, Fu G, Gindulyte A, Han L, He J, He S, Shoemaker BA, Wang J, Yu B, Zhang J, Bryant SH. PubChem Substance and Compound Databases. ***Nucleic Acids Res.,*** *44,* D1202-13 (2016).
173. Berman HM, Westbrook J, Feng Z, Gilliland G, Bhat TN, Weissig H, Shindyalov IN, Bourne PE. The Protein Data Bank. ***Nucleic Acids Res.,*** *28,* 235-42 (2000).
174. Yin J, Henriksen NM, Slochower DR, Shirts MR, Chiu MW, Mobley DL, Gilson MK. Overview of the SAMPL5 Host-Guest Challenge: Are We Doing Better? ***J Comput. Aid. Mol. Des.,*** *31,* 1-19 (2017).
175. Feenstra KA, Hess B, Berendsen HJC. Improving Efficiency of Large Time-Scale Molecular Dynamics Simulations of Hydrogen-Rich Systems. ***J. Comput. Chem.,*** *20,* 786-96 (1999).
176. Gresh N. Development, Validation, and Applications of Anisotropic Polarizable Molecular Mechanics to Study Ligand and Drug-Receptor Interactions. ***Curr. Pharm. Design,*** *12,* 2121-58 (2006).
177. Cisneros GA, Piquemal J-P, Darden TA. Generalization of the Gaussian Electrostatic Model: Extension to Arbitrary Angular Momentum, Distributed Multipoles, and Speedup with Reciprocal Space Methods. ***J. Chem. Phys.,*** *125,* 184101 (2006).
178. Rackers JA, Wang Q, Liu C, Piquemal J-P, Ren P, Ponder JW. An Optimized Charge Penetration Model for Use with the AMOEBA Force Field. ***Phys. Chem. Chem. Phys.,*** *19,* 276-91 (2017).
179. Narth C, Lagardère L, Polack E, Gresh N, Wang Q, Bell DR, Rackers JA, Ponder JW, Ren PY, Piquemal J-P. Scalable Improvement of SPME Multipolar Electrostatics in Anisotropic Polarizable Molecular Mechanics Using a General Short‐Range Penetration Correction Up to Quadrupoles. ***J. Comput. Chem.,*** *37,* 494-506 (2016).

180. Wang Q, Rackers JA, He C, Qi R, Narth C, Lagardere L, Gresh N, Ponder JW, Piquemal J-P, Ren P. General Model for Treating Short-Range Electrostatic Penetration in a Molecular Mechanics Force Dield. ***J. Chem. Theory Comput.,*** *11,* 2609-18 (2015).
181. Lipparini F, Lagardère L, Stamm B, Cancès E, Schnieders M, Ren P, Maday Y, Piquemal J-P. Scalable Evaluation of Polarization Energy and Associated Forces in Polarizable Molecular Dynamics: I. Toward Massively Parallel Direct Space Computations. ***J. Chem. Theory Comput.,*** *10,* 1638-51 (2014).
182. Lagardère L, Lipparini F, Polack E, Stamm B, Cancès E, Schnieders M, Ren P, Maday Y, Piquemal J-P. Scalable Evaluation of Polarization Energy and Associated Forces in Polarizable Molecular Dynamics: II. Toward Massively Parallel Computations Using Smooth Particle Mesh Ewald. ***J. Chem. Theory Comput.,*** *11,* 2589-99 (2015).

Table 1. Tinker 8 File Name Suffixes and Descriptions

SUFFIX	Description of File Contents
.xyz	Cartesian coordinates, atom types and connectivity
.int	Internal coordinates as a Z-matrix
.mol	MDL MOL structure compatible with Tinker
.mol2	MOL2 structure compatible with Tinker
.pdb	PDB structure compatible with Tinker
.arc	Structure archive, e.g., MD trajectory
.dyn	MD restart information
.hes	Cartesian Hessian matrix
.key	Control file with Tinker keywords
tinker.key	Generic keyfile
.err	Current structure at error occurance
.seq	Biopolymer sequence
.vel	Atomic velocities
.ind	Atomic induced dipole moments
.dma	Distributed multipole values
.bar	Window energy values for BAR and FEP
.prm	Force field parameter file
.doc	Detailed parameter descriptions
.end	Requests orderly termination of Tinker program
.vb1, .vb2, .blk	Block iterative vibrational mode files
.001, .002, *etc.*	"Cycle" files containing sequential structure output

Table 2. Tinker 8 CPU and Tinker-OpenMM GPU MD simulation timings in ns/day. [a]

			CPU [b]	GPU [c]		
SYSTEM	**POTENTIAL**	**ATOMS**		**970**	**1070**	**1080Ti**
WaterSmall	AMOEBA	648	4.78	61.6	98.4	125.9
WaterBox	TIP3P	1500	14.2	361.7	574.9	671.9
Crambin	AMOEBA	1920	1.12	43.0	64.2	72.0
CBClip	AMOEBA	6432	0.664	20.9	32.5	46.1
DHFR	AMOEBA	23558	0.164	8.62	13.1	20.0
DHFR	Amber ff99SB	23558	1.16	78.4	115.1	204.7
COX-2	AMOEBA	174219	0.0176	1.05	1.67	2.27
COX-2	Amber ff99SB	174219	0.150	10.7	15.3	24.6

[a] All simulations run with 2 fs MD time steps; RESPA integrator and OPT polarization model for AMOEBA, Verlet integrator with constraints used to enforce rigid water and fixed bond lengths to hydrogen for TIP3P and Amber ff99SB potentials.
[b] Apple Mac Pro with an Intel 6-Core Xeon E5650 Processor running at 2.66 GHz.
[c] NVIDIA Maxwell and Pascal Series GTX GPU cards, run via Tinker-OpenMM.

Figure Captions

Figure 1. Diagram showing the main component programs of the Tinker 8 package, organized into eight functional classes.

Figure 2. A schematic procedure illustrating how to construct a new Tinker program.

Figure 3. Binding free energy calculation for the model system cucurbit[7]uril and 3-amino-1-adamantanol. (A) Structures of host and guest, (B) Predicted binding pose from *dynamic,* (C) Experimental and predicted binding free energy.

Figure 4. Structural optimization of Deca-Alanine in the gas phase using (A) the *scan* program and (B) the *monte* program.

Figure 5. A typical thermodynamic cycle for the calculation of absolute binding free energy of host and ligand in Tinker. The completely solvated ligand and a solvent box are associated through intermediate states with gradual changes in the order parameters of vdW and electrostatics. While the order parameter of electrostatics affects both intermolecular and intramolecular interactions, the decreasing order parameter of vdW only decouples the ligand from the environment and does not change the intramolecular vdW interaction. A restraint is added to prevent the possible bad contacts and to help sampling.

Figure 6. Force Field Explorer (FFE) displaying the Dickerson dodecamer structure of B-form DNA. The expandable tree structure in the left panel provides access to coordinate and type information at the molecule, residue and atom levels.

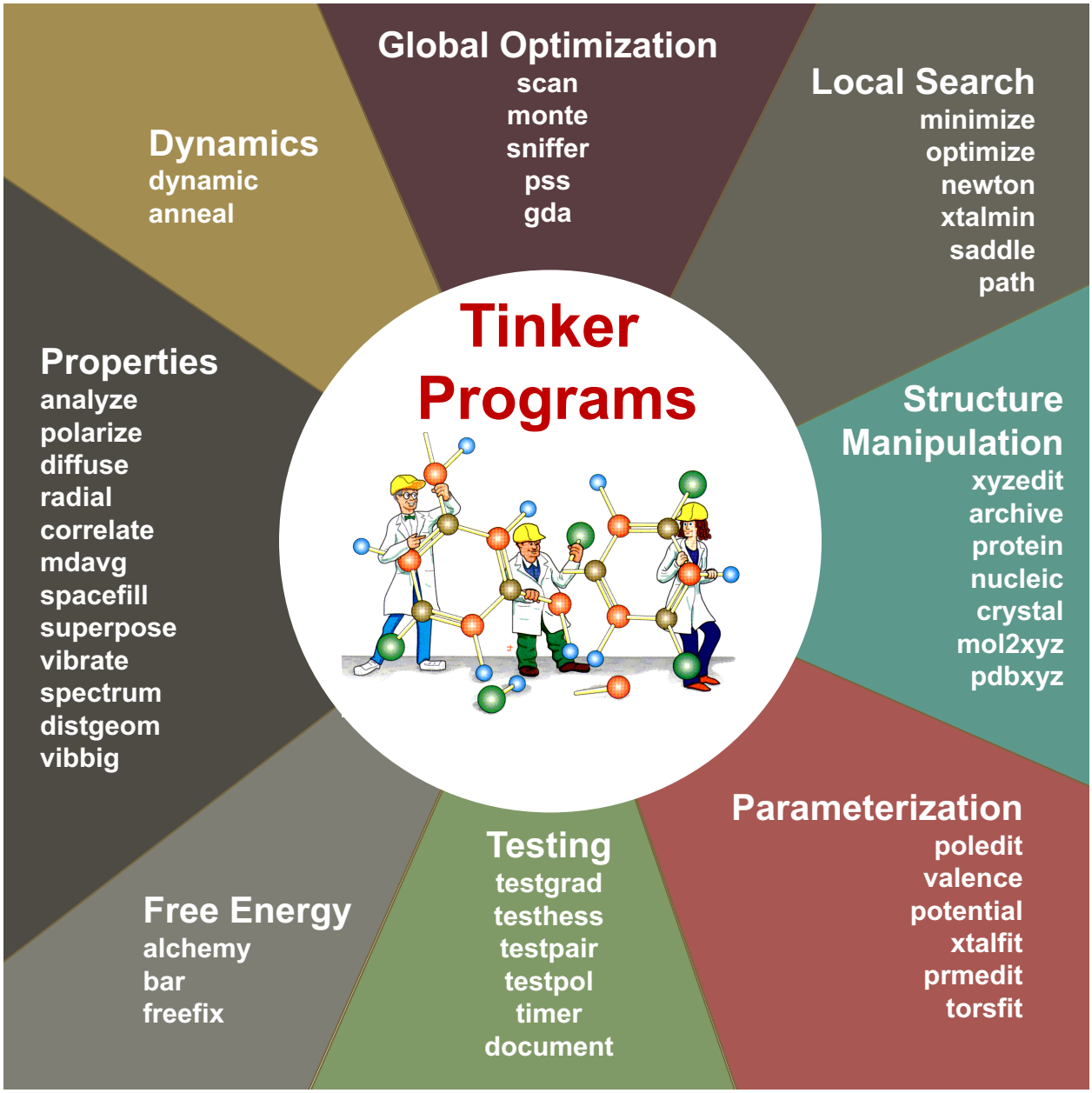

Global Optimization
scan
monte
sniffer
pss
gda
Local Search
minimize
optimize
newton
xtalmin
saddle
path
Dynamics
dynamic
anneal
Tinker Programs
Properties
analyze
polarize
diffuse
radial
correlate
mdavg
spacefill
superpose
vibrate
spectrum
distgeom
vibbig
Structure Manipulation
xyzedit
archive
protein
nucleic
crystal
mol2xyz
pdbxyz
Parameterization
poledit
valence
potential
xtalfit
prmedit
torsfit
Testing
testgrad
testhess
testpair
testpol
timer
document
Free Energy
alchemy
bar
freefix

Figure 1

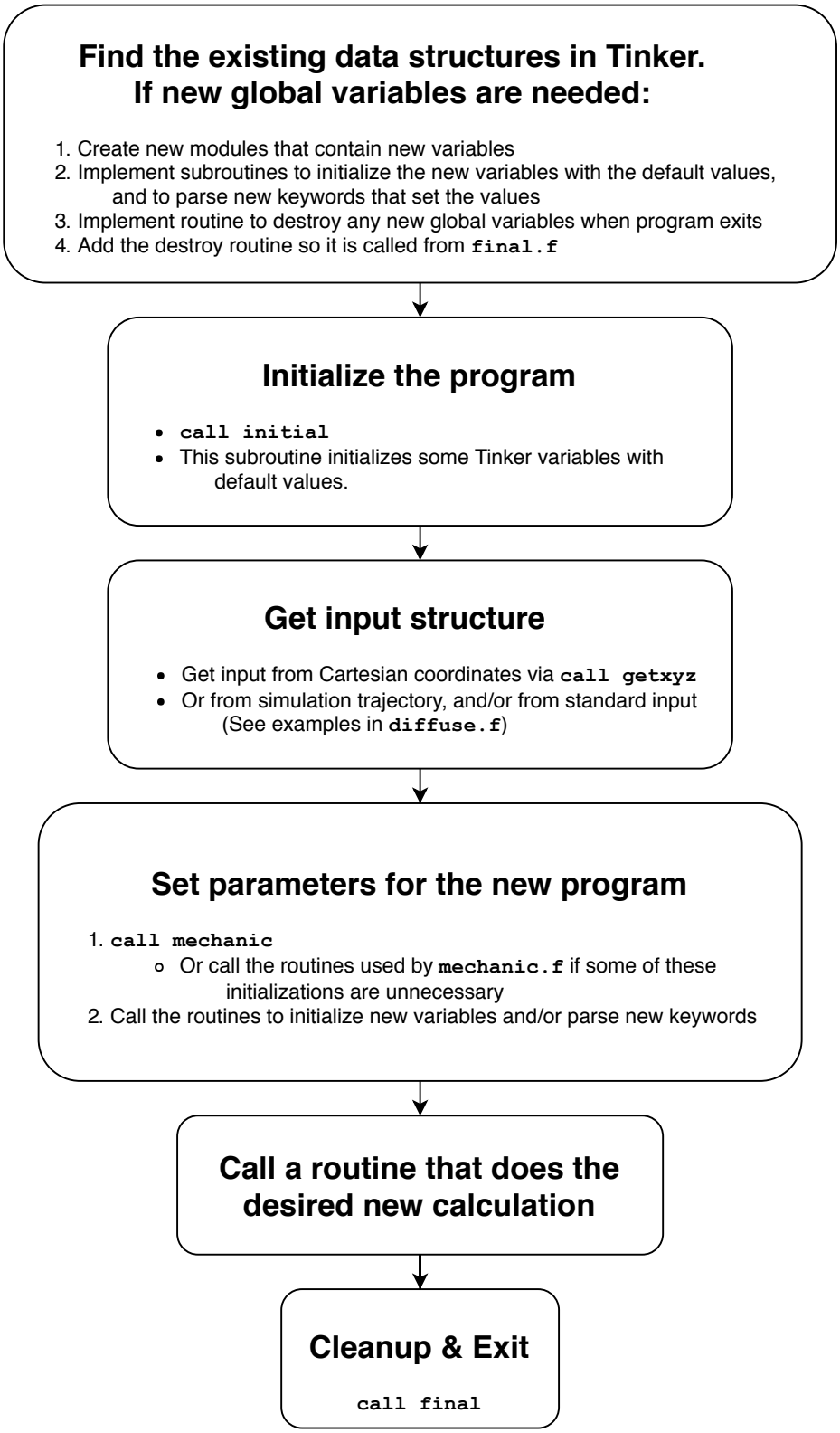
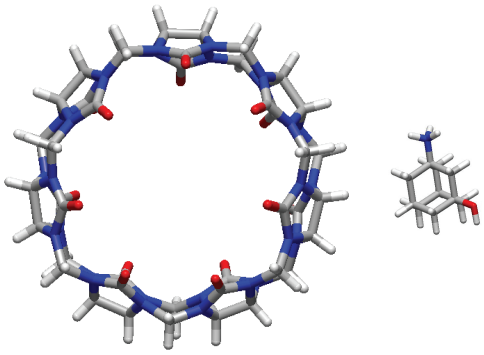
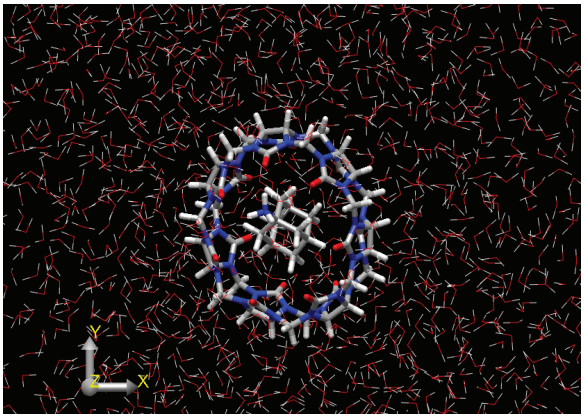

Find the existing data structures in Tinker.
If new global variables are needed:
1. Create new modules that contain new variables
2. Implement subroutines to initialize the new variables with the default values, and to parse new keywords that set the values
3. Implement routine to destroy any new global variables when program exits
4. Add the destroy routine so it is called from final.f
Initialize the program
call initial
This subroutine initializes some Tinker variables with default values.
Get input structure
Get input from Cartesian coordinates via call getxyz
Or from simulation trajectory, and/or from standard input (See examples in diffuse.f)
Set parameters for the new program
1. call mechanic
Or call the routines used by mechanic.f if some of these initializations are unnecessary
2. Call the routines to initialize new variables and/or parse new keywords
Call a routine that does the desired new calculation
Cleanup & Exit
call final

Figure 2

A)



B)



C)

	Experimental	Predicted
Binding Free Energy (kcal/mol)	−11.6 ± 0.1	−12.41± 0.7

Figure 3

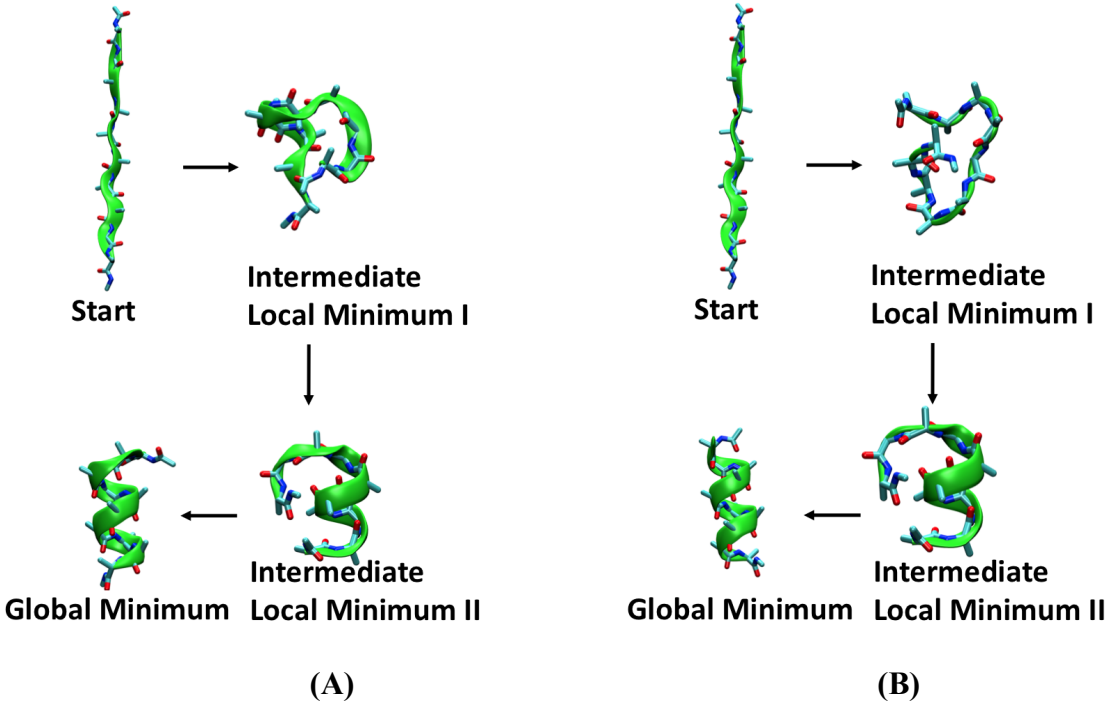

Start
Intermediate
Local Minimum I
Intermediate
Local Minimum II
Global Minimum
(A)
Start
Intermediate
Local Minimum I
Intermediate
Local Minimum II
Global Minimum
(B)

Figure 4

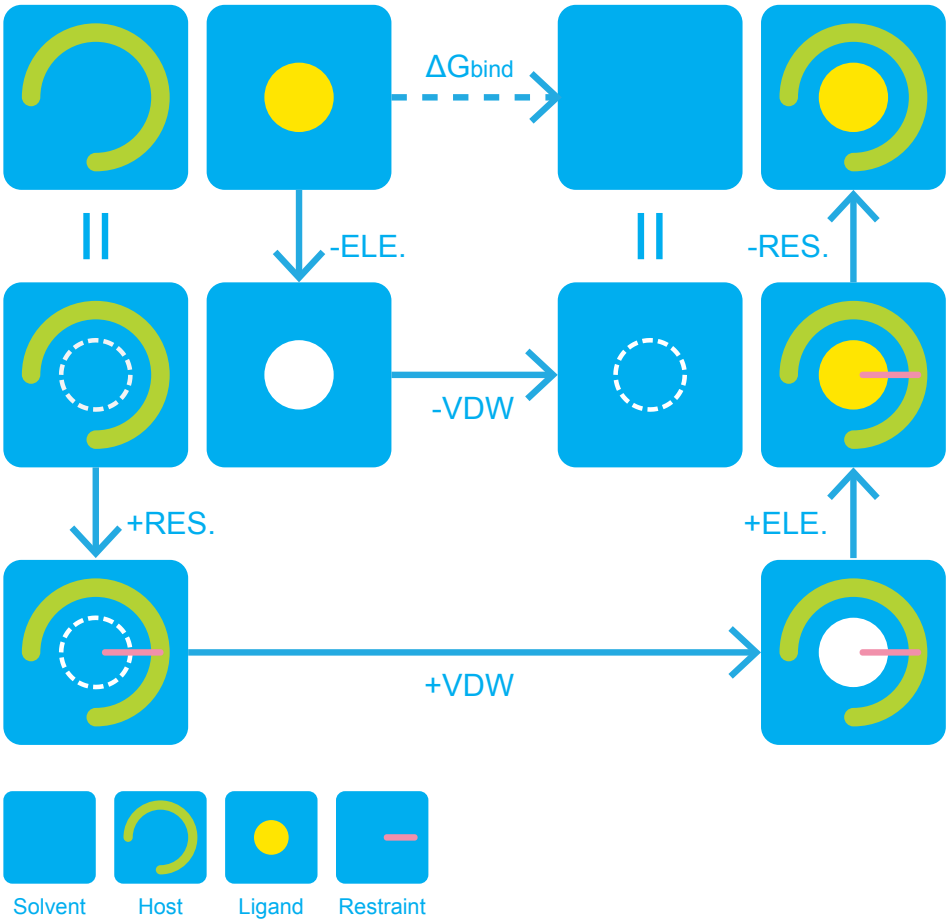

ΔGbind
-ELE.
-RES.
-VDW
+RES.
+ELE.
+VDW
Solvent
Host
Ligand
Restraint

Figure 5

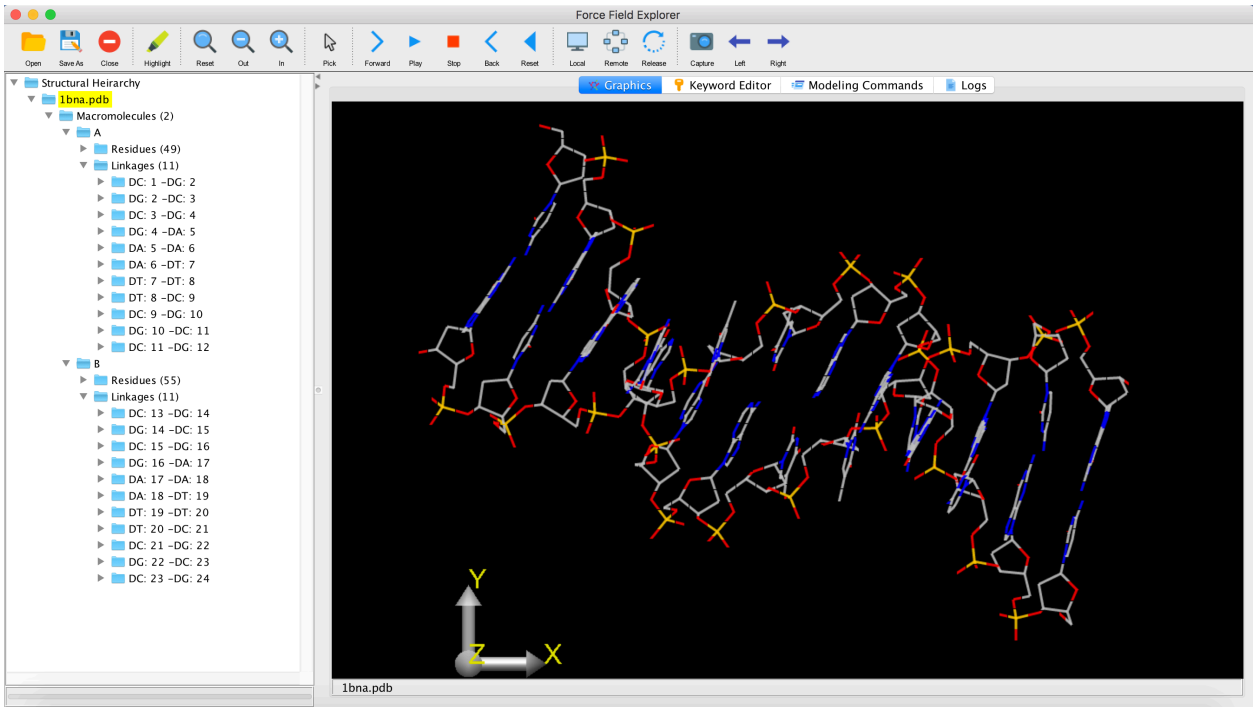

Force Field Explorer
Open
Save As
Close
Highlight
Reset
Out
In
Pick
Forward
Play
Stop
Back
Reset
Local
Remote
Release
Capture
Left
Right
Structural Heirarchy
1bna.pdb
Macromolecules (2)
A
Residues (49)
Linkages (11)
DC: 1 –DG: 2
DG: 2 –DC: 3
DC: 3 –DG: 4
DG: 4 –DA: 5
DA: 5 –DA: 6
DA: 6 –DT: 7
DT: 7 –DT: 8
DT: 8 –DC: 9
DC: 9 –DG: 10
DG: 10 –DC: 11
DC: 11 –DG: 12
B
Residues (55)
Linkages (11)
DC: 13 –DG: 14
DG: 14 –DC: 15
DC: 15 –DG: 16
DG: 16 –DA: 17
DA: 17 –DA: 18
DA: 18 –DT: 19
DT: 19 –DT: 20
DT: 20 –DC: 21
DC: 21 –DG: 22
DG: 22 –DC: 23
DC: 23 –DG: 24
Graphics
Keyword Editor
Modeling Commands
Logs
Y
Z
X
1bna.pdb

Figure 6