JCTC Journal of Chemical Theory and Computation

Article

Cite This: J. Chem. Theory Comput. 2018, 14, 5273 5289

pubs.acs.org/JCTC

Tinker 8: Software Tools for Molecular Design

Joshua A. Rackers,[†] Zhi Wang,[‡] Chao Lu,[‡] Marie L. Laury,[‡] Louis Lagardère,[§] Michael J. Schnieders,[∥] Jean-Philip Piquemal,[§] Pengyu Ren,[⊥] and Jay W. Ponder*[,†,‡]

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

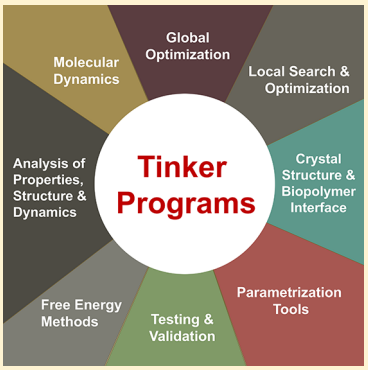
[‡]Department of Chemistry, Washington University in Saint Louis, Saint Louis, Missouri 63130, United States

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

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

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

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 polarizable models such as the Atomic Multipole Optimized Energetics for Biomolecular Applications (AMOEBA) force field. 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 message passing interface (MPI) parallel distributed memory supercomputers and 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 mechanics (MM) calculations and molecular dynamics (MD) simulations, with special emphasis on biomolecular computations. 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 message passing interface (MPI) parallelizaton (Tinker HP),[1] and for graphical processing unit (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] and GitHub repositories.[4]

Tinker originated as a new software package implementing the MM2[5] and MM3[6] force fields of Allinger for use in conformational analysis of organic natural products.[7] An early prototype of the software was incorporated as the basis of molecular mechanics calculations in the ChemOffice software package.[8] Additional applications used this early pre Tinker platform for the development of efficient structure optimization algorithms for large molecules[9] and for packing analysis of amino acid side chains in folded protein structures.[10] Development under the name Tinker began in earnest at Washington University in the mid 1990s, 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 incorporation of existing empirical potentials as well as design and parametrization 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,[11] 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,[12] PyMOL,[13] Jmol,[14] Force Field X,[15] Open Babel,[16] MDTraj,[17] MDAnalysis,[18] ParmEd,[19] Molden,[20] VEGA ZZ,[21] PACKMOL,[22] ForceBalance,[23] WebMO,[24] and many others. Access to Tinker, including the Atomic Multipole Optimized Energetics for Biomolecular Applications (AMOEBA) polarizable multipole force field, is also available from the CHARMM modeling software via the MSCALE interface facility.[25]

The current Tinker 8 package contains roughly 60 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

Received: May 31, 2018
Published: September 3, 2018

DOI: 10.1021/acs.jctc.8b00529
J. Chem. Theory Comput. 2018, 14, 5273−5289

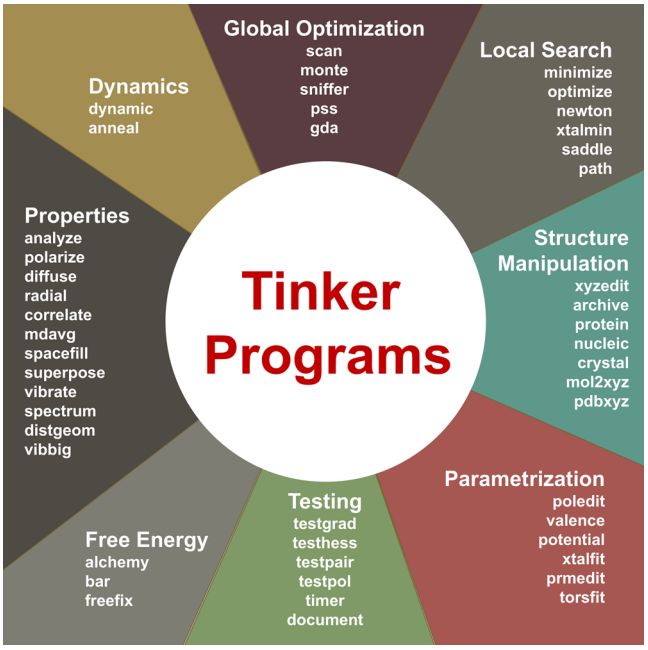


Figure 1. Diagram showing the main component programs of the Tinker 8 package, organized into eight functional classes.

computations are performed in full double precision arithmetic. The only hard limits on program size are the allowed total number of 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 1 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 usable 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 computing 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[26] 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 application programming interface (API). It provides a *dynamic omm* program that exchanges data between CPU and GPU memory through the library interface and performs MD simulations on CUDA compatible NVIDIA GPUs. Tinker OpenMM supports an increasing subset of Tinker's energy functions, MD 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 the capability to run 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 names of 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.

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 occurrence
.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

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 is written 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 (α, β, and γ). The standard convention used in Tinker places the **a** lattice vector along the global x axis and the **b** vector in the xy plane. These periodic dimensions are stored as part of the keyword control (*.key*) file for a calculation or, optionally, as part of the coordinate file itself. Periodic systems, including truncated octahedra, are defined such that the centroid of the unit cell or periodic box is located at the coordinate origin $(0, 0, 0)$.

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 system wide resources are managed in Fortran modules that make use of dynamic memory allocation and are designed to represent only 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.

DOI: 10.1021/acs.jctc.8b00529
J. Chem. Theory Comput. 2018, 14, 5273−5289

The central component of the Tinker package is a modular set of callable routines that (1) manage the package owned resources, including default initialization, allocation of the dynamic memory, release of the allocated space, etc., (2) perform MM calculations and MD simulations on a single set of parameters and atomic coordinates, (3) read in settings from standard input, command line arguments, and external files and write out the current state of the system to standard output or external files. These routines essentially work as the underlying API to build the higher level routines and programs in the Tinker package. For example, the *gradient* routine is called not only 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 reversible reference system propagator algorithm (RESPA) integrator, for which the energy and force terms are organized into "fast" and "slow" groups that are evaluated on different time scales. Because these energy and force routines are organized as a callable library, RESPA is integrated at a high level simply by 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 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 Set Up a Macromolecular Simulation. One of the most common use cases for Tinker is running MD simulations on a macromolecular system of interest in explicit solvent. Setting up this kind of calculation requires a starting set of coordinates that includes the macromolecule, the solvent, and, in many cases, relevant ions. Tinker contains all of the tools needed to create this starting set of coordinates. Below we will briefly outline the tools and how to use them. While it should be noted that there are multiple software packages capable of performing all or most of the following steps, the purpose of this tutorial is to show that this workflow can be accomplished entirely within the Tinker suite of programs.

The following steps outline how to set up a simulation of a macromolecule in explicit water with ions starting from a PDB file.

1. Obtain coordinates for the macromolecule.
    a. Download a PDB file for the molecule of interest.
    b. Use the *pdbxyz* program to convert the PDB file to a Tinker *.xyz* file and select the desired force field model.
2. Create a box of water that is large enough to contain the macromolecule.
    a. Copy a Tinker *.xyz* file for water into the working directory (one can be found in the /example directory of the Tinker distribution).
    b. Use the *xyzedit* program and select the option to create and fill a periodic boundary box with specified dimensions.
3. Place the macromolecule in the solvent box.
    a. Use *xyzedit* and select the option to place the macromolecule structure into the solvent box.
    b. (optional) Alleviate any bad contacts by running the Tinker *minimize* program on the resulting structure.
4. Place ions around the macromolecule
    a. Use *xyzedit* and select the option to place specified positive and negative ions around the macromolecule.
    b. Use the Tinker *analyze* program with option M to check that the total charge of the system is neutral.

These steps, followed appropriately, yield a structure of a macromolecule of interest in explicit solvent and ions that can be used to start a simulation. However, this procedure supplies only a relatively rough set of starting coordinates. The user must choose the force field model with which to simulate the system and must equilibrate the system by running a short MD trajectory. The type of model and necessary length of equilibration are left to the discretion of the user.

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

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

These steps, which are 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 does not require any new global variables, this can be done via the *initial* and *getxyz* routines. The *initial* routine declares and initializes global variable values that are needed for every Tinker program, and *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 have been called, Tinker is ready to perform operations on the structure. Multistructure "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 MM 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, *mechanic* 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 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

DOI: 10.1021/acs.jctc.8b00529
J. Chem. Theory Comput. 2018, 14, 5273−5289

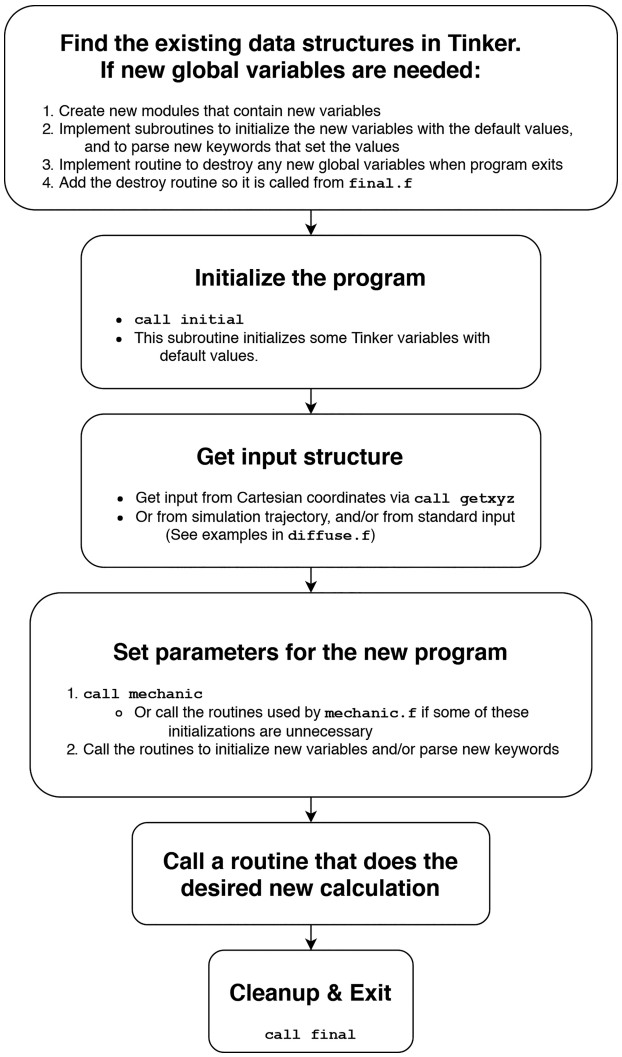


Figure 2. Schematic procedure illustrating construction of a Tinker program.

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 a new potential energy function is also straightforward. The developer simply adds the code for the function to the pre existing, 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. At this point, Tinker is 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 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 provide users the ability to explore a wide variety of models. 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. Various Tinker potential terms can be grouped together to replicate several widely used biomolecular force fields such as those from the CHARMM,[27] Amber,[28] and OPLS AA[29] fami lies. The second reason to support a large number of poten tials is to expedite the development of new models. Because of the modular nature of the code, researchers can easily incor porate any of the existing potentials in a model. In total, approximately 30 different potential terms are supported in the Tinker package, all with exact analytical energies and Cartesian derivatives and many with second derivatives. Broadly, the poten tials 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 couplings between the primary energy terms. The simplest of the primary terms are the bonded potentials. Tinker includes harmonic, anharmonic, and Morse bond terms. The package also has several types of angle bending potentials: harmonic, anharmonic, linear, projected in plane, and Fourier based angles. Additionally, four types of torsion terms are included. The first is a calculation for a simple torsion defined by four consecutively bonded atoms using a sum of Fourier terms. The second, termed a Bell's "π torsion", computes the torsion around a bond connecting two trigonal centers using the π orbital directions at each trigonal center.[30] Tinker also includes so called "improper torsion" terms that define torsions involving atoms that are not consecutively bonded, 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. The final primary potential term in Tinker 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[31] or Allinger[32] defini tions. 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 constitute the bulk of most intramolecular energy models a user might like to build or use.

In addition to primary intramolecular potentials, Tinker 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

DOI: 10.1021/acs.jctc.8b00529
J. Chem. Theory Comput. 2018, 14, 5273−5289

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 term.[33] 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,[34] a buffered 14−7 Halgren potential,[35] a Buckingham exponential−6 potential,[36] a Gaussian vdW potential, and the MM3 vdW−hydrogen bond potential.[37] Distance based vdW cutoffs combined with pair neighbor lists are available to avoid computation of N^2 interactions in large systems. A long range correction is available for all vdW potentials using a mean field approach to include contributions to the energy and internal virial from the cutoff distance to infinity.[38] This correction is highly accurate for homogeneous systems but less appropriate for systems with vdW heterogeneity. 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 advanced models, such as the AMOEBA force field.[39] 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,[40] ACE,[41] and Grycuk;[42] the generalized Kirkwood (GK)[43] method for use with polarizable multipoles; accessible surface area based solvation;[44] the hydrophobic potential of mean force (HPMF),[45] a novel reaction field method;[46] and Poisson−Boltzmann (PB)[47] 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.[48] 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 selected quantum effects within a classical framework. Simple π orbital calculations of the Hückel, Pariser−Parr−Pople, or variable electronegativity self consistent field (VESCF)[49] 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.[50]

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 that they can each be called separately with just one line of computer code. For developers, this means that it is easy to mix and match different potentials in a model or devise new potentials as desired. For users, this makes it simple 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 for 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,[51] CHARMM,[52] OPLS,[53] MM2/3,[5,54] MMFF,[55] AMOEBA,[39b−d,56] Dang,[57] 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 coordinate files 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 with other formats. For instance, CHARMM, AMBER, and VMD are adapted to PDB files, several pharmaceutical modeling suites and drug databases use MOL2 files, and many quantum mechanics (QM) packages such as Gaussian operate on internal coordinates. To allow interoperability, Tinker provides six commands to do interconversions 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 a *.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.

DOI: 10.1021/acs.jctc.8b00529
J. Chem. Theory Comput. 2018, 14, 5273−5289

Second, Tinker also provides file editing tools for the purpose of simulation setup. Most of the *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 an optimal root mean square deviation (RMSD) using a noniterative quaternion based algorithm.[58] Since biomolecules such as nucleic acids and proteins are target systems for many studies, Tinker provides the *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 *crystal* utility is designed for manipulation of crystal structures, including generation of 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) minimization,[59] optimally conditioned variable metric (OCVM) nonlinear optimization,[60] and truncated Newton conjugate gradient (TNCG)[9,61] 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 close to the final transition state. Tinker contains two other methods, *saddle* and *path*, that are specifically designed to locate conformational transition states and pathways. The *saddle* routine represents a combination of ideas from the Halgren−Lipscomb synchronous transit[62] and Bell−Crighton quadratic path[63] methods. It takes two end point 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.[64]

In addition, Tinker contains an adaptive derivative free multidimensional 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.[65] 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, and 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,[66] generalized gradient descent,[67] "jumping between wells",[68] and the Monte Carlo minimization (MCM) method.[69] The second category of global optimization algorithms includes potential smoothing techniques[70] and the related Gaussian density annealing (GDA) scheme.[71]

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 overcome. Then the cooling schedule is applied to gradually lower the temperature and coalesce the structure 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.[67,72] The *scan* program uses jumping between wells to locate all of 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 of the local minima.[68] The *monte* program implements an MCM protocol that uses Metropolis Monte Carlo exploration of a potential surface in which the energy of each point on the surface is remapped to the value of the closest local minimum.[69a] Potential surface smoothing (PSS) views the original potential energy functional forms as the $t = 0$ initial conditions for solution of the diffusion equation. Conformational search is then performed on the smoother surface produced at some finite nonzero 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.

DOI: 10.1021/acs.jctc.8b00529
J. Chem. Theory Comput. 2018, 14, 5273−5289

Two examples of global optimization methods are demonstrated in Figure 3 for a deca alanine model system in

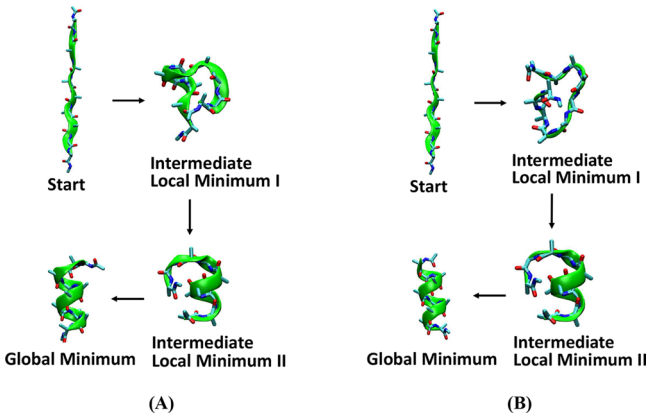


Figure 3. Structural optimization of deca alanine in the gas phase using (A) the *scan* program and (B) the *monte* program.

the gas phase using the *scan* and *monte* programs. The two 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 3. 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 3. Though they follow different paths in moving around the surface, the two methods appear to produce similar partially optimized structures, shown as intermediate local minima II in Figure 3.

Dynamics Methods. One important feature for any modern molecular mechanics software package is the ability to perform molecular dynamics. 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: microcanonical (*NVE*), cannonical (*NVT*), isenthalpic−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,[73] stochastic,[74] Nosé−Hoover *NPT*,[75] Bussi−Parrinello *NPT*,[76] a two stage, multiple time step RESPA integrator,[77] and a rigid body integrator.[78] 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, frequently evaluated time step is used for rapidly changing degrees of freedom such as bond stretching, and the second, longer time step is used for the slowly 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[79] in order to implement holonomic constraints within velocity Verlet and related integrators. In addition, Tinker contains a stochastic dynamics integrator[80] that employs a series expansion to treat small frictional coefficients[81] and has the ability to scale the friction term based on accessible surface area.[82] Removal of translation and, if appropriate, rotation of the global system is optionally invoked after each user specified number of MD steps.

For the constant temperature and constant pressure ensembles, Tinker includes a variety of thermostats and barostats. The included thermostats are Bussi,[83] Berendsen,[84] Andersen[85] and Nosé−Hoover.[75,86] The available barostats are Berendsen,[84] Bussi−Parrinello,[76] and Monte Carlo.[87] 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 the application of an isokinetic scheme that combines a massive thermostat with a multiple time step integrator to achieve ultralong time steps for the slowly evolving but computationally expensive potential terms in a simulation. This method is called 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.[88]

Properties and Analysis. One of the 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 it 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 principal axes. It calculates the internal virial, numerical, and virial based derivatives of the energy with respect to volume, and finally, 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 multiframe archive or MD trajectory as a Tinker *.arc* file. These features allow users not only 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 magnitudes 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 multiframe 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

DOI: 10.1021/acs.jctc.8b00529
J. Chem. Theory Comput. 2018, 14, 5273−5289

an iterative sliding block diagonalization method that finds the lowest frequencies and corresponding modes with $O(N^2)$ computational effort.[89]

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.

The *correlate* routine is 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. Tinker contains methods to compute the binding free energy of a drug to a protein or the solvation free energy of an ion in water. Computation of binding free energies relies on the completion of a thermodynamic cycle, as pictured in Figure 4. 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 (FEP).

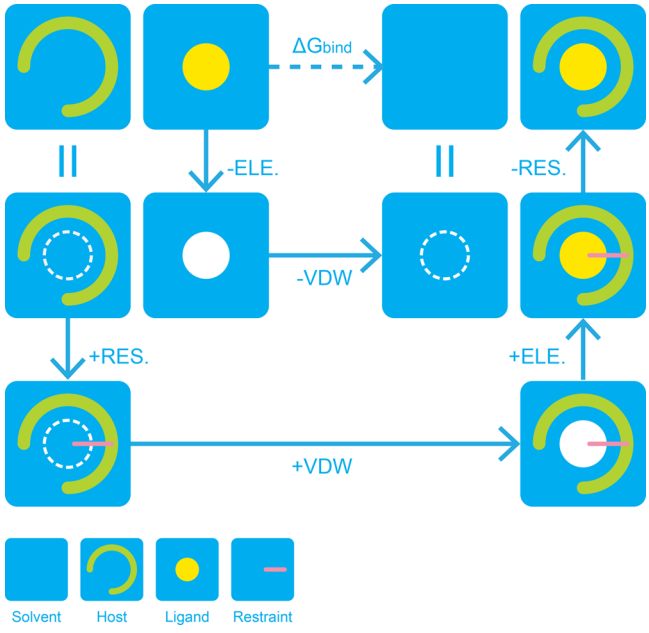


Figure 4. Typical thermodynamic cycle for the calculation of the absolute binding free energy of a host and ligand in Tinker. The completely solvated ligand and a solvent box are associated through intermediate states with gradual changes in the order parameters of vdW and electrostatics. While the order 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 possible bad contacts and to help with sampling.

The majority of the analysis of the free energy difference of the sampled conformations in Tinker is handled by the *bar* program, which applies the standard Zwanzig FEP method[90] and Bennett acceptance ratio (BAR) method[91] for the canonical ensemble. Additionally, the *bar* program has been extended to process isothermal−isobaric simulations[92] and to estimate the differences in entropy and enthalpy of the samples.[93]

An example of the utility of the *dynamic* and *bar* programs is the calculation of binding free energies for the SAMPL4 host−guest challenge.[94] 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* FEP calculations on those trajectories. The results for one particular host−guest pair are shown in Figure 5. 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 into them. Tinker has many built in utilities to test the correctness of code for new and existing models. These allow developers to quickly test whether a new energy function and its derivatives are consistent. The *testgrad* and *testrot* programs check to make sure that 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 the next step by comparing the analytical Hessian against one computed numerically from either gradient or potential energy values. 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.

Additionally, 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 allowing comparison with experimental values, computing the molecular polarizability gives users an idea of how strongly many body effects may affect subsequent calculations.

Parametrization Tools. The final set of important utilities in Tinker are a trio of programs designed to parametrize 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.[95] It is also used to set local coordinate frames for atomic multipoles, 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 to evaluate and refine atomic multipole models. This utility computes the

DOI: 10.1021/acs.jctc.8b00529
J. Chem. Theory Comput. 2018, 14, 5273−5289

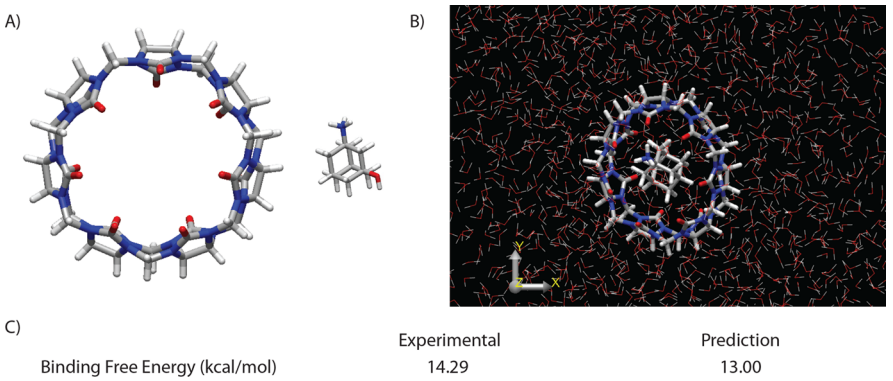


Figure 5. Binding free energy calculation for the model system cucurbit[7]uril and 3 amino 1 adamantanol. (A) Structures of the host and guest. (B) Predicted binding pose from *dynamic*. (C) Experimental and predicted binding free energies.

electrostatic potential on a grid of points 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 parametrization programs are combined in a Python based, publicly available software package called Poltype.[96] 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 parametrization 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 algo rithms to help address computational bottlenecks, including algorithms 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: orthogonal, monoclinic, triclinic, and octahedral. 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 dis tances and list buffers can be configured separately through keywords for the van der Waals interactions, the partial charges, the atomic multipoles, and the polarization preconditioner, respectively, to speed neighbor searching, as opposed to the naive 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[97] 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 electro statics 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 for multipolar models written by Thomas Darden, the PME routine in Tinker 8 gives the user control over the Ewald damping parameter and allows the use of either "tinfoil" or vacuum boundary condi tions. The PME module also supports the truncated octa hedron as a periodic shape and allows PME calculations to be performed on nonperiodic systems. The current Tinker imple mentation closely follows the multipole PME version previously described by Sagui et al.[98] 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 is otherwise rate limiting for large systems. Tinker optionally uses either a refactored three dimensional (3D) ver sion of the public domain FFTPACK Fourier transform code or the fast Fourier transform package FFTW (Fastest Fourier Transform in the West)[99] 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

$$\boldsymbol{\mu}_i = \alpha_i \mathbf{F}_i$$

where $\boldsymbol{\mu}$, α, and $\mathbf{F}$ 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:

$$\mathbf{F}_i = \mathbf{F}_i^{\text{perm}} + \mathbf{F}_i^{\text{ind}}$$

This gives rise to the total induction energy,

$$U^{\text{ind}} = \frac{1}{2}\sum_i \boldsymbol{\mu}_i \cdot \mathbf{F}_i^{\text{ind}} - \sum_i \boldsymbol{\mu}_i \cdot \mathbf{F}_i^{\text{perm}}$$

where all that is needed is to solve for the induced dipoles of the system. Tinker has three methods for determining the induced dipoles of a system: preconditioned conjugate gradient (PCG), optimized perturbation theory (OPT), and extended Lagrangian/self consistent field (iEL SCF).

DOI: 10.1021/acs.jctc.8b00529
J. Chem. Theory Comput. 2018, 14, 5273−5289

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

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

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 SCF calculation. In Tinker this is done using a PCG algorithm,[100] which is typically able to converge the calculation within five or six iterations.

The OPT method[101] 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,

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

where each order of the perturbation is determined by

$$\boldsymbol{\mu}_0 = \alpha\mathbf{F}^{perm}$$

$$\lambda\boldsymbol{\mu}_1 = \lambda\alpha\mathbf{F}^{ind(\boldsymbol{\mu}_0)}$$

$$\lambda^2\boldsymbol{\mu}_2 = \lambda^2\alpha\mathbf{F}^{ind(\boldsymbol{\mu}_1)}$$

$$\vdots$$

$$\lambda^n\boldsymbol{\mu}_n = \lambda^n\alpha\mathbf{F}^{ind(\boldsymbol{\mu}_{n-1})}$$

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

$$U = \sum_i \boldsymbol{\mu}_i^{OPT}\cdot\mathbf{F}_i^{perm}$$

$$\boldsymbol{\mu}^{OPT} = M_0\boldsymbol{\mu}_0 + M_1\boldsymbol{\mu}_1 + M_2\boldsymbol{\mu}_2 + ... + M_n\boldsymbol{\mu}_n$$

where the coefficients M_i 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 two to four terms is a reasonable approximation that gives a speed boost over traditional PCG.

The final method included with Tinker is the iEL SCF method.[102] This method minimizes the number of iterations needed in solving the induced dipoles by introducing the Lagrangian,

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

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

$$m_i\ddot{\mathbf{r}}_i = -\frac{\partial U_{AMOEBA}(\mathbf{r}^N, \boldsymbol{\mu}_{SCF}^N)}{\partial\mathbf{r}_i}$$

$$\boldsymbol{\mu}_i = \omega^2(\boldsymbol{\mu}_{SCF,i} - \boldsymbol{\mu}_i)$$

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, later versions of Tinker 8 include two additional polarization options. The first is an extension of the iEL SCF method called iEL 0SCF.[103] This method employs the same auxiliary dipoles as in the iEL SCF scheme, but they are used to drive the dynamics directly instead of being used as a starting point for SCF. 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, previously incorporated into the Tinker HP code base, is the truncated conjugate gradient (TCG) method.[104] 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 the need for any parameters such as those 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. Like the OPT method described above, the TCG method provides a fully analytical set of induced dipoles that approximate the fully converged SCF values.

Orthogonal-Space Random Walk. Besides the typical 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" simulta neously.[105] Because of the complexity of many systems, efficient sampling of the hidden degrees of freedom dominates the accuracy of the 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 soft core modified buffered 14−7 potential is applied as a replacement for the original. Permanent electrostatic interactions are also modified by a soft core treatment to prevent numerical instability during simulation.[106] When OSRW is used with AMOEBA, the polarization energy and forces are computed using an interpolation between fully charged/polarizable and de charged/nonpolarizable ligand atoms as described previ ously.[107] 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.[108]

The setup of a Tinker keyfile for the use of OSRW is straightforward. For instance, to compute the hydration free energy of a 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", "donoligand condensed", and "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 a method for generating a structure or structures consistent with an input set of distance

DOI: 10.1021/acs.jctc.8b00529
J. Chem. Theory Comput. 2018, 14, 5273−5289

constraints.[109,110] A basic DG algorithm takes an object in a high dimensional mathematical "distance space" and reduces the 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 nuclear Overhauser effect (NOE) distance constraints.[111] However, a more interesting application of DG is to underconstrained problems. Given a limited set of upper and lower bound distances between atoms or groups in a molecular system, one would like a DG 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 underconstrained 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.[112] Tinker uses a powerful but relatively little known shortest path update algorithm to further reduce the metrization workload.[113] 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.[114] 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[115] and provides good sampling in less constrained situations such as protein structure prediction.[116]

6. FORCE FIELD EXPLORER

In addition to the suite of command line programs, Tinker 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,[117] the NCI CACTUS database, or the RCSB Protein Data Bank (PDB).[118] Biopolymers can also be interactively constructed from sequences in various idealized structures. The program also gives users the ability to play back any Tinker MD trajectory. 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 the 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 GUI, 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 keyfile via a graphical editing facility. With access to the keyfile, users can quickly and easily change the options for whatever calculation they are running without touching the command line. As shown in the example in Figure 6, FFE's functionality is laid out in an easy to navigate format. Combined with the full integration of Tinker, this 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 174 219 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,[119] a solvated DHFR protein, and a solvated COX 2 protein dimer. The system sizes differ by more than 2 orders of magnitude. The force fields tested were Amber ff99SB[51e] and AMOEBA. All of the simulations were performed with a 2 fs MD time step, and throughputs (in nanoseconds per day) are reported in Table 2. The CPU based Tinker calculations are performed in full double precision arithmetic. The GPU results use the "mixed" precision mode available in OpenMM, whereby energies and forces are single precision, while MD integration steps are double precision.

We note that hydrogen mass reweighting,[120] which retards high frequency motions, is a keyword option available in Tinker. Use of this option coupled with tight thermostating enables stable MD trajectories with 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 AND 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. Three major projects are 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

DOI: 10.1021/acs.jctc.8b00529
J. Chem. Theory Comput. 2018, 14, 5273−5289

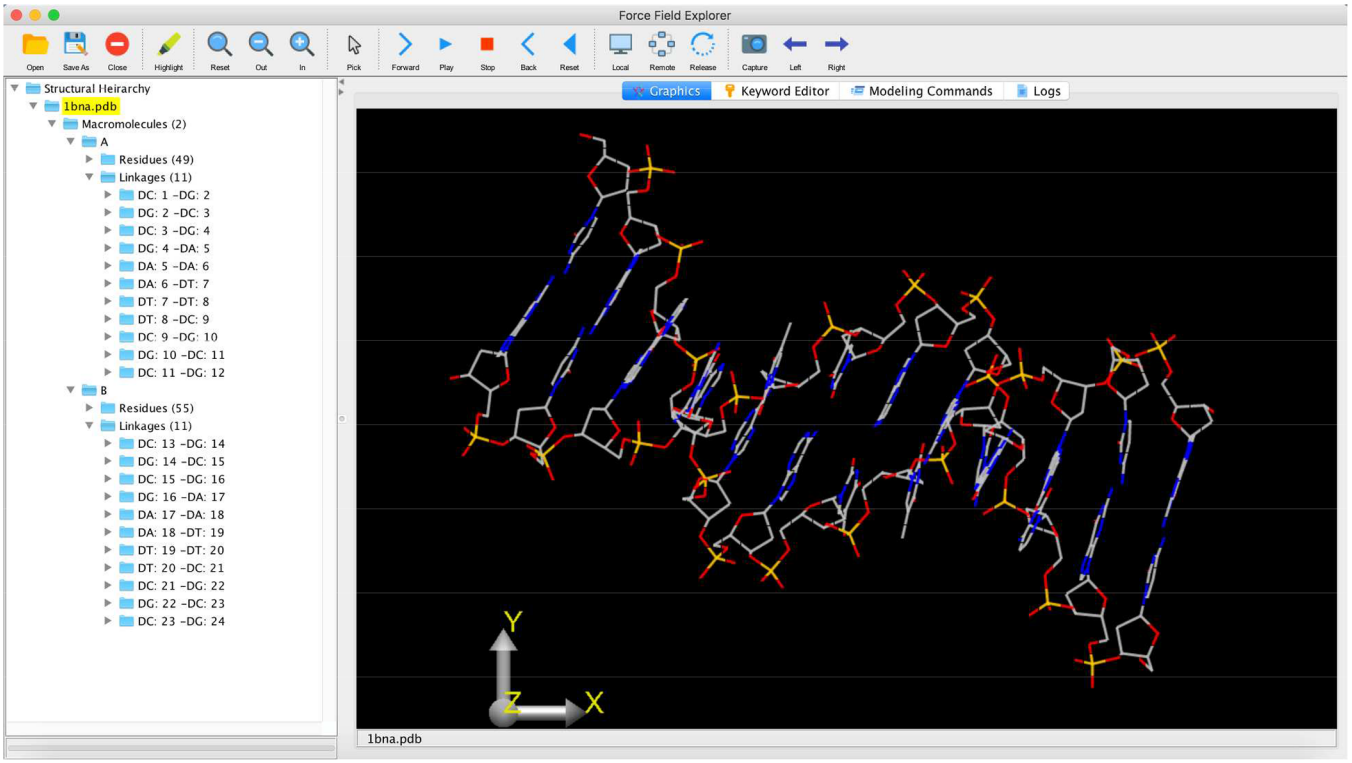


Figure 6. Force Field Explorer (FFE) displaying the Dickerson dodecamer structure of B form DNA. The expandable tree structure in the left panel provides access to coordinate and type information at the molecule, residue, and atom levels.

Table 2. Tinker 8 CPU and Tinker OpenMM GPU MD Simulation Timings (in ns/day)[a]

			CPU[b]	GPU[c]				
system	potential	no. of atoms	E5650	GT 750m	GTX 970	GTX 1070	GTX 1080Ti	GV100
WaterSmall	AMOEBA	648	4.78	12.6	61.6	98.4	125.9	121.2
WaterBox	TIP3P	1500	14.2	99.7	361.7	574.9	671.9	495.3
Crambin	AMOEBA	1920	1.12	4.46	43.0	64.2	72.0	84.0
CBClip	AMOEBA	6432	0.664	1.27	20.9	32.5	46.1	48.8
DHFR	AMOEBA	23558	0.164	0.497	8.62	13.1	20.0	24.9
DHFR	Amber ff99SB	23558	1.16	8.85	78.4	115.1	204.7	219.6
COX-2	AMOEBA	174219	0.0176	0.0652	1.05	1.67	2.27	3.70
COX-2	Amber ff99SB	174219	0.150	1.18	10.7	15.3	24.6	43.0

[a]All of the simulations were run with periodic boundary conditions, PME electrostatics, and 2 fs MD time steps. The RESPA integrator and OPT polarization were used for AMOEBA, and the Verlet integrator with constrained rigid water and fixed bonds to hydrogen was used for TIP3P and Amber ff99SB. [b]Tinker 8 in double precision mode on an Intel six core Xeon E5650 processor at 2.66 GHz. [c]Tinker OpenMM in mixed precision mode on several NVIDIA GPU cards.

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 up the costliest aspect of polarizable force fields, solution of the polarization model itself, are under development for imple mentation in future versions of Tinker, as is support for current polarizable models including SIBFA[121] and GEM.[122]

In addition to efficient software for existing force fields, the Tinker project is developing code that will run the next gen eration of models. A new class of physics based potentials are under development that rely less on empiricism than their predecessors. These models attempt to correct for errors that occur at short range in point charge 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 recently been published,[123] 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 model ing packages (Tinker, Tinker HP,[124] 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 platforms, from large scale CPU based supercomputers to individual GPUs. The Tinker HP branch for massively parallel

DOI: 10.1021/acs.jctc.8b00529
J. Chem. Theory Comput. 2018, 14, 5273−5289

CPU calculations and the Tinker OpenMM branch as a CUDA based GPU implementation are responsible for enabling this high performance. A goal of the Tinker project is to unify the code structures of these software packages. This will have 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 hope to ensure users 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 that allows 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 the development of tools that make the Tinker toolbox useful.

AUTHOR INFORMATION

Corresponding Author

*E mail: ponder@dasher.wustl.edu.

ORCID

Jean-Philip Piquemal: 0000 0001 6615 9426

Jay W. Ponder: 0000 0001 5450 9230

Funding

J.W.P. and P.R. thank the National Institutes of Health, National Institute of General Medical Sciences for support of recent force field and software development via Awards R01 GM106137 and R01 GM114237. J.A.R. thanks the NSF Molecular Sciences Software Institute for support via a MolSSI Software Fellowship.

Notes

The authors declare no competing financial interest.

ACKNOWLEDGMENTS

As with most large software packages under development for many years, Tinker has a very large number of contributors—far too many to list here—who have provided code and suggestions. J.W.P. 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, Z. F.; Harger, M.; Torabifard, H.; Cisneros, G. A.; Schnieders, M. J.; Gresh, N.; Maday, Y.; Ren, P. Y.; Ponder, J. W.; 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.* **2018**, *9*, 956−972.

(2) Harger, M.; Li, D.; Wang, Z.; Dalby, K.; Lagardère, L.; Piquemal, J. P.; Ponder, J. W.; Ren, P. Y. Tinker OpenMM: Absolute and Relative Alchemical Free Energies Using AMOEBA on GPUs. *J. Comput. Chem.* **2017**, *38*, 2047−2055.

(3) Ponder, J. W. Tinker Molecular Modeling. https://dasher.wustl.edu/tinker/ (accessed July 23, 2018). Piquemal, J. P. Piquemal Research & Software. http://piquemalresearch.com/research and softwares/ (accessed July 23, 2018). Ren, P. Tinker GPU Main Page. http://biomol.bme.utexas.edu/tinkergpu/ (accessed July 23, 2018).

(4) Ponder, J. W. Tinker: Software Tools for Molecular Design. https://github.com/TinkerTools/tinker (accessed July 23, 2018). Ren, P. Tinker OpenMM Toolkit for Molecular Simulation Using High Performance GPU Code. https://github.com/pren/tinkeropenmm/ (accessed July 23, 2018).

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

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

(7) Corey, E. J.; Ponder, J. W. Stereochemistry of the Hygrolidins. *Tetrahedron Lett.* **1984**, *25*, 4325−4328.

(8) *ChemOffice*; CambridgeSoft: Cambridge, MA, 2018.

(9) Ponder, J. W.; Richards, F. M. An Efficient Newton like Method for Molecular Mechanics Energy Minimization of Large Molecules. *J. Comput. Chem.* **1987**, *8*, 1016−1024.

(10) Ponder, J. W.; Richards, F. M. Tertiary Templates for Proteins: Use of Packing Criteria in the Enumeration of Allowed Sequences for Different Structural Classes. *J. Mol. Biol.* **1987**, *193*, 775−791.

(11) Pande, V. S.; Baker, I.; Chapman, J.; Elmer, S. P.; Khaliq, S.; Larson, S. M.; Rhee, Y. M.; Shirts, M. R.; Snow, C. D.; Sorin, E. J.; Zagrovic, B. Atomistic Protein Folding Simulations on the Submillisecond Time Scale Using Worldwide Distributed Computing. *Biopolymers* **2003**, *68*, 91−109.

(12) Humphrey, W.; Dalke, A.; Schulten, K. VMD Visual Molecular Dynamics. *J. Mol. Graphics* **1996**, *14*, 33−38.

(13) DeLano, W. L. PyMOL: An Open Source Molecular Graphics Tool. *CCP4 Newsl. Protein Crystallogr.* **2002**, *40*, 82−92.

(14) Hanson, R. M. Jmol A Paradigm Shift in Crystallographic Visualization. *J. Appl. Crystallogr.* **2010**, *43*, 1250−1260.

(15) LuCore, S. D.; Litman, J. M.; Powers, K. T.; Gao, S.; Lynn, A. M.; Tollefson, W. T. A.; Fenn, T. D.; Washington, M. T.; Schnieders, M. J. Dead End Elimination with a Polarizable Force Field Repacks PCNA Structures. *Biophys. J.* **2015**, *109*, 816−826.

(16) O'Boyle, N. M.; Banck, M.; James, C. A.; Morley, C.; Vandermeersch, T.; Hutchison, G. R. Open Babel: An Open Chemical Toolbox. *J. Cheminf.* **2011**, *3*, 33.

(17) McGibbon, R. T.; Beauchamp, K. A.; Harrigan, M. P.; Klein, C.; Swails, J.; Hernandez, C. X.; Schwantes, C. R.; Wang, L. P.; Lane, T. J.; Pande, V. S. MDTraj: A Modern Open Library for the Analysis of Molecular Dynamics Trajectories. *Biophys. J.* **2015**, *109*, 1528−1532.

(18) Michaud Agrawal, M.; Denning, E. J.; Woolf, T. B.; Beckstein, O. MDAnalysis: A Toolkit for the Analysis of Molecular Dynamics Simulations. *J. Comput. Chem.* **2011**, *32*, 2319−2327.

(19) Swails, J.; Hernandez, C.; Mobley, D.; Nguyen, H.; Wang, L. P.; Janowski, P. ParmEd: Parameter/Topology Editor and Molecular Simulator. https://github.com/ParmEd/ParmEd (accessed July 23, 2018).

(20) Schaftenaar, G.; Vlieg, E.; Vriend, G. Molden 2.0: Quantum Chemistry Meets Proteins. *J. Comput. Aided Mol. Des.* **2017**, *31*, 789−800.

(21) 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. Aided Mol. Des.* **2004**, *18*, 167−173.

(22) Martinez, L.; Andrade, R.; Birgin, E. G.; Martinez, J. M. Packmol: A Package for Building Initiial Configurations for Molecular Dynamics Simulations. *J. Comput. Chem.* **2009**, *30*, 2157−2164.

(23) Wang, L. P.; Martinez, T. J.; Pande, V. S. Building Force Fields: An Automatic, Systematic, and Reproducible Approach. *J. Phys. Chem. Lett.* **2014**, *5*, 1885−1891.

DOI: 10.1021/acs.jctc.8b00529
J. Chem. Theory Comput. 2018, 14, 5273−5289

(24) Schmidt, J. R.; Polik, W. F. *WebMO Enterprise*, version 13.0; WebMO LLC: Holland, MI, 2013.
(25) Woodcock, H. L.; Miller, B. T.; Hodoscek, M.; Okur, A.; Larkin, J. D.; Ponder, J. W.; Brooks, B. R. MSCALE: A General Utility for Multiscale Modeling. *J. Chem. Theory Comput.* **2011**, *7*, 1208−1219.
(26) (a) Eastman, P.; Friedrichs, M. S.; Chodera, J. D.; Radmer, R. J.; Bruns, C. M.; Ku, J. P.; Beauchamp, K. A.; Lane, T. J.; Wang, L. P.; Shukla, D.; Tye, T.; Houston, M.; Stich, T.; Klein, C.; Shirts, M. R.; Pande, V. S. OpenMM 4: A Reusable, Extensible, Hardware Independent Library for High Performance Molecular Simulation. *J. Chem. Theory Comput.* **2013**, *9*, 461−469. (b) Eastman, P.; Swails, J.; Chodera, J. D.; McGibbon, R. T.; Zhao, Y.; Beauchamp, K. A.; Wang, L. P.; Simmonett, A. C.; Harrigan, M. P.; Stern, C. D.; Wiewiora, R. P.; Brooks, B. R.; Pande, V. S. OpenMM 7: Rapid Development of High Performance Algorithms for Molecular Dynamics. *PLoS Comput. Biol.* **2017**, *13*, e1005659.
(27) Huang, J.; Rauscher, S.; Nawrocki, G.; Ran, T.; Feig, M.; de Groot, B. L.; Grubmüller, H.; MacKerell, A. D., Jr. CHARMM36m: An Improved Force Field for Folded and Intrinsically Disordered Proteins. *Nat. Methods* **2017**, *14*, 71−73.
(28) Maier, J. A.; Martinez, C.; Kasavajhala, K.; Wickstrom, L.; Hauser, K. E.; Simmerling, C. ff14SB: Improving the Accuracy of Protein Side Chain and Backbone Parameters from ff99SB. *J. Chem. Theory Comput.* **2015**, *11*, 3696−3713.
(29) Robertson, M. J.; Tirado Rives, J.; Jorgensen, W. L. Improved Peptide and Protein Torsional energetics with the OPLS AA Force Field. *J. Chem. Theory Comput.* **2015**, *11*, 3499−3509.
(30) Palmo, K.; Mannfors, B.; Mirkin, N. G.; Krimm, S. Potential Energy Functions: From Consistent Force Fields to Spectroscopically Determined Polarizable Force Fields. *Biopolymers* **2003**, *68*, 383−394.
(31) Wilson, E. B., Jr.; Decius, J. G.; Cross, P. G.; Lagemann, R. T. Molecular Vibration. *Am. J. Phys.* **1955**, *23*, 550−550.
(32) Liljefors, T.; Tai, J. C.; Li, S.; Allinger, N. L. On the Out of Plane Deformation of Aromatic Rings, and Its Representation by Molecular Mechanics. *J. Comput. Chem.* **1987**, *8*, 1051−1056.
(33) Urey, H. C.; Bradley, C. A., Jr. The Vibrations of Pentatonic Tetrahedral Molecules. *Phys. Rev.* **1931**, *38*, 1969−1978.
(34) Lennard Jones, J. E. Cohesion. *P. Phys. Soc.* **1931**, *43*, 461−482.
(35) Halgren, T. A. Representation of van der Waals (vdW) Interactions in Molecular Mechanics Force Fields: Potential Form, Combination Rules, and vdW Parameters. *J. Am. Chem. Soc.* **1992**, *114*, 7827−7843.
(36) Buckingham, R. A. The Classical Equation of State of Gaseous Helium, Neon and Argon. *Proc. R. Soc. London, Ser. A* **1938**, *168*, 264−283.
(37) (a) Lii, J. H.; Allinger, N. L. Directional Hydrogen Bonding in the MM3 Force Field. *J. Phys. Org. Chem.* **1994**, *7*, 591−609. (b) Lii, J. H.; Allinger, N. L. Directional Hydrogen Bonding in the MM3 Force Field: II. *J. Comput. Chem.* **1998**, *19*, 1001−1016.
(38) (a) Allen, M. P.; Tildesley, D. H. *Computer Simulation of Liquids*; Oxford University Press: New York, 1987. (b) Shirts, M. R.; Mobley, D.; Chodera, J. D.; Pande, V. S. Accurate and Efficient Corrections for Missing Dispersion Interactions in Molecular Simulations. *J. Phys. Chem. B* **2007**, *111*, 13052−13063.
(39) (a) Ponder, J. W.; Wu, C.; Ren, P.; Pande, V. S.; Chodera, J. D.; Schnieders, M. J.; Haque, I.; Mobley, D. L.; Lambrecht, D. S.; DiStasio, R. A., Jr.; Head Gordon, M.; Clark, G. N. I.; Johnson, M. E.; Head Gordon, T. Current Status of the AMOEBA Polarizable Force Field. *J. Phys. Chem. B* **2010**, *114*, 2549−2564. (b) Ren, P.; Ponder, J. W. Polarizable Atomic Multipole Water Model for Molecular Mechanics Simulation. *J. Phys. Chem. B* **2003**, *107*, 5933−5947. (c) Ren, P.; Wu, C.; Ponder, J. W. Polarizable Atomic Multipole based Molecular Mechanics for Organic Molecules. *J. Chem. Theory Comput.* **2011**, *7*, 3143−3161. (d) Shi, Y.; Xia, Z.; Zhang, J.; Best, R.; Wu, C.; Ponder, J. W.; Ren, P. Polarizable Atomic Multipole based AMOEBA Force Field for Proteins. *J. Chem. Theory Comput.* **2013**, *9*, 4046−4063. (e) Zhang, C.; Lu, C.; Jing, Z.; Wu, C.; Piquemal, J. P.; Ponder, J. W.; Ren, P. AMOEBA Polarizable Atomic Multipole Force Field for Nucleic Acids. *J. Chem. Theory Comput.* **2018**, *14*, 2084−2108.
(40) Onufriev, A.; Case, D. A.; Bashford, D. Effective Born Radii in the Generalized Born Approximation: The Importance of Being Perfect. *J. Comput. Chem.* **2002**, *23*, 1297−1304.
(41) 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.* **2001**, *22*, 1857−1879.
(42) Grycuk, T. Deficiency of the Coulomb Field Approximation in the Generalized Born Model: An Improved Formula for Born Radii Evaluation. *J. Chem. Phys.* **2003**, *119*, 4817−4826.
(43) Schnieders, M. J.; Ponder, J. W. Polarizable Atomic Multipole Solutes in a Generalized Kirkwood Continuum. *J. Chem. Theory Comput.* **2007**, *3*, 2083−2097.
(44) Wesson, L.; Eisenberg, D. Atomic Solvation Parameters Applied to Molecular Dynamics of Proteins in Solution. *Protein Sci.* **1992**, *1*, 227−235.
(45) Lin, M. S.; Fawzi, N. L.; Head Gordon, T. Hydrophobic Potential of Mean Force as a Solvation Function for Protein Structure Prediction. *Structure* **2007**, *15*, 727−740.
(46) Kong, Y.; Ponder, J. W. Calculation of the Reaction Field due to Off Center Point Multipoles. *J. Chem. Phys.* **1997**, *107*, 481−492.
(47) (a) Warwicker, J.; Watson, H. C. Calculation of the Electric Potential in the Active Site Cleft due to α Helix Dipoles. *J. Mol. Biol.* **1982**, *157*, 671−679. (b) 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: Struct., Funct., Genet.* **1986**, *1*, 47−59. (c) Sharp, K. A.; Honig, B. Electrostatic Interactions in Macromolecules: Theory and Applications. *Annu. Rev. Biophys. Biophys. Chem.* **1990**, *19*, 301−332.
(48) (a) Baker, N. A.; Sept, D.; Joseph, S.; Holst, M. J.; McCammon, J. A. Electrostatics of Nanosystems: Application to Microtubules and the Ribosome. *Proc. Natl. Acad. Sci. U. S. A.* **2001**, *98*, 10037−10041. (b) Schnieders, M. J.; Baker, N. A.; Ren, P.; Ponder, J. W. Polarizable Atomic Multipole Solutes in a Poisson−Boltzmann Continuum. *J. Chem. Phys.* **2007**, *126*, 124114.
(49) Allinger, N. L.; Li, F.; Yan, L.; Tai, J. C. Molecular Mechanics (MM3) Calculations on Conjugated Hydrocarbons. *J. Comput. Chem.* **1990**, *11*, 868−895.
(50) (a) Xiang, J. Y.; Ponder, J. W. A Valence Bond Model for Aqueous Cu(II) and Zn(II) Ions in the AMOEBA Polarizable Force Field. *J. Comput. Chem.* **2013**, *34*, 739−749. (b) Xiang, J. Y.; Ponder, J. W. An Angular Overlap Model for Cu(II) Ion in the AMOEBA Polarizable Force Field. *J. Chem. Theory Comput.* **2014**, *10*, 298−311. (c) Carlsson, A. E.; Zapata, S. The Functional Form of Angular Forces around Transition Metal Ions in Biomolecules. *Biophys. J.* **2001**, *81*, 1−10.
(51) (a) Cornell, W. D.; Cieplak, P.; Bayly, C. I.; Gould, I. R.; Merz, K. M.; Ferguson, D. M.; Spellmeyer, D. C.; Fox, T.; Caldwell, J. W.; Kollman, P. A. A Second Generation Force Field for the Simulation of Proteins, Nucleic Acids, and Organic Molecules. *J. Am. Chem. Soc.* **1995**, *117*, 5179−5197. (b) 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. In *Computer Simulation of Biomolecular Systems*; van Gunsteren, W. F., Weiner, P. K., Wilkinson, A. J., Eds.; Springer: Dordrecht, The Netherlands, 1997; Vol. 3, pp 83−96. (c) Cheatham, T. E., III; Cieplak, P.; Kollman, P. A. A Modified Version of the Cornell et al. Force Field with Improved Sugar Pucker Phases and Helical Repeat. *J. Biomol. Struct. Dyn.* **1999**, *16*, 845−862. (d) Wang, J.; Cieplak, P.; Kollman, P. A. How Well Does a Restrained Electrostatic Potential (RESP) Model Perform in Calculating Conformational Energies of Organic and Biological Molecules? *J. Comput. Chem.* **2000**, *21*, 1049−1074. (e) Hornak, V.; Abel, R.; Okur, A.; Strockbine, B.; Roitberg, A.; Simmerling, C. Comparison of Multiple Amber Force Fields and

DOI: 10.1021/acs.jctc.8b00529
J. Chem. Theory Comput. 2018, 14, 5273−5289

Development of Improved Protein Backbone Parameters. *Proteins: Struct., Funct., Genet.* **2006**, *65*, 712−725.
(52) (a) Neria, E.; Fischer, S.; Karplus, M. Simulation of Activation Free Energies in Molecular Systems. *J. Chem. Phys.* **1996**, *105*, 1902−1921. (b) MacKerell, A. D., Jr.; Bashford, D.; Bellott, M. L. D. R.; Dunbrack, R. L., Jr.; Evanseck, J. D.; Field, M. J.; Fischer, S.; Gao, J.; Guo, H.; Ha, S.; et al. All Atom Empirical Potential for Molecular Modeling and Dynamics Studies of Proteins. *J. Phys. Chem. B* **1998**, *102*, 3586−3616. (c) Foloppe, N.; MacKerell, A. D., Jr. All Atom Empirical Force Field for Nucleic Acids: I. Parameter Optimization Based on Small Molecule and Condensed Phase Macromolecular Target Data. *J. Comput. Chem.* **2000**, *21*, 86−104. (d) MacKerell, A. D., Jr.; Feig, M.; Brooks, C. L., III 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.* **2004**, *25*, 1400−1415.
(53) (a) Jorgensen, W. L.; Maxwell, D. S.; 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.* **1996**, *118*, 11225−11236. (b) Kaminski, G. A.; Friesner, R. A.; Tirado Rives, J.; Jorgensen, W. L. Evaluation and Reparamet rization of the OPLS AA Force Field for Proteins via Comparison with Accurate Quantum Chemical Calculations on Peptides. *J. Phys. Chem. B* **2001**, *105*, 6474−6487. (c) Weiner, S. J.; Kollman, P. A.; Case, D. A.; Singh, U. C.; 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.* **1984**, *106*, 765−784. (d) Jorgensen, W. L.; Severance, D. L. Aromatic Aromatic Interactions: Free Energy Profiles for the Benzene Dimer in Water, Chloroform, and Liquid Benzene. *J. Am. Chem. Soc.* **1990**, *112*, 4768−4774. (e) Maxwell, D. S.; Tirado Rives, J.; Jorgensen, W. L. 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.* **1995**, *16*, 984−1010. (f) Jorgensen, W. L.; 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.* **1988**, *110*, 1657−1666.
(54) (a) Sprague, J. T.; Tai, J. C.; Yuh, Y. H.; Allinger, N. L. The MMP2 Calculational Method. *J. Comput. Chem.* **1987**, *8*, 581−603. (b) Allinger, N. L.; Kok, R. A.; Imam, M. R. Hydrogen Bonding in MM2. *J. Comput. Chem.* **1988**, *9*, 591−595. (c) Lii, J. H.; Allinger, N. L. 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.* **1989**, *111*, 8576−8582. (d) Allinger, N. L.; Li, F.; Yan, L. Molecular Mechanics. The MM3 Force Field for Alkenes. *J. Comput. Chem.* **1990**, *11*, 848−867. (e) Lii, J. H.; Allinger, N. L. The MM3 Force Field for Amides, Polypeptides and Proteins. *J. Comput. Chem.* **1991**, *12*, 186−199.
(55) Halgren, T. A.; Nachbar, R. B. Merck Molecular Force Field. IV. Conformational Energies and Geometries for MMFF94. *J. Comput. Chem.* **1996**, *17*, 587−615.
(56) (a) Ren, P.; Ponder, J. W. Consistent Treatment of Inter and Intramolecular Polarization in Molecular Mechanics Calculations. *J. Comput. Chem.* **2002**, *23*, 1497−1506. (b) Wu, J. C.; 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.* **2010**, *6*, 2059−2070. (c) Grossfield, A.; Ren, P.; Ponder, J. W. Ion Solvation Thermodynamics from Simulation with a Polarizable Force Field. *J. Am. Chem. Soc.* **2003**, *125*, 15671−15682.
(57) (a) Dang, L. X. Development of Nonadditive Intermolecular Potentials Using Molecular Dynamics: Solvation of Li+ and F− Ions in Polarizable Water. *J. Chem. Phys.* **1992**, *96*, 6970−6977. (b) Smith, D. E.; Dang, L. X. Interionic Potentials of Mean Force for SrCl2 in Polarizable Water: A Computer Simulation Study. *Chem. Phys. Lett.* **1994**, *230*, 209−214. (c) Dang, L. X.; Chang, T. M. Molecular Dynamics Study of Water Clusters, Liquid, and Liquid−Vapor Interface of Water with Many Body Potentials. *J. Chem. Phys.* **1997**, *106*, 8149−8159. (d) Chang, T. M.; Dang, L. X. Detailed Study of Potassium Solvation Using Molecular Dynamics Techniques. *J. Phys. Chem. B* **1999**, *103*, 4714−4720. (e) Dang, L. X. Intermolecular Interactions of Liquid Dichloromethane and Equilibrium Properties of Liquid−Vapor and Liquid−Liquid Interfaces: A Molecular Dynamics Study. *J. Chem. Phys.* **1999**, *110*, 10113−10122. (f) Chang, T. M.; Dang, L. X. On Rotational Dynamics of an NH4+ Ion in Water. *J. Chem. Phys.* **2003**, *118*, 8813−8820. (g) Dang, L. X.; Schenter, G. K.; Glezakou, V. A.; Fulton, J. L. Molecular Simulation Analysis and X Ray Absorption Measurement of Ca2+, K+ and Cl Ions in Solution. *J. Phys. Chem. B* **2006**, *110*, 23644−23654. (h) Wick, C. D.; Dang, L. X. Molecular Dynamics Study of Ion Transfer and Distribution at the Interface of Water and 1, 2 Dichlorethane. *J. Phys. Chem. C* **2008**, *112*, 647−649. (i) Sun, X.; Chang, T. M.; Cao, Y.; Niwayama, S.; Hase, W. L.; Dang, L. X. Solvation of Dimethyl Succinate in a Sodium Hydroxide Aqueous Solution. A Computational Study. *J. Phys. Chem. B* **2009**, *113*, 6473−6477. (j) Dang, L. X.; Truong, T. B.; Ginovska Pangovska, B. Note: Interionic Potentials of Mean Force for Ca2+ Cl− in Polarizable Water. *J. Chem. Phys.* **2012**, *136*, 126101.
(58) Kearsley, S. K. On the Orthogonal Transformation Used for Structural Comparisons. *Acta Crystallogr., Sect. A: Found. Crystallogr.* **1989**, *45*, 208−210.
(59) (a) Liu, D. C.; Nocedal, J. On the Limited Memory BFGS Method for Large Scale Optimization. *Math. Program.* **1989**, *45*, 503−528. (b) Nocedal, J. Updating Quasi Newton Matrices with Limited Storage. *Math. Comput.* **1980**, *35*, 773−782. (c) Nocedal, J.; Wright, S. J. *Numerical Optimization*, 2nd ed.; Springer: New York, 2006.
(60) (a) Shanno, D. F.; Phua, K. H. Numerical Comparison of Several Variable Metric Algorithms. *J. Optimiz. Theory Appl.* **1978**, *25*, 507−518. (b) Davidon, W. C. Optimally Conditioned Optimization Algorithms without Line Searches. *Math. Program.* **1975**, *9*, 1−30.
(61) Dembo, R. S.; Steihaug, T. Truncated Newton Algorithms for Large Scale Unconstrained Optimization. *Math. Program.* **1983**, *26*, 190−212.
(62) (a) Halgren, T. A.; Lipscomb, W. N. The Synchronous Transit Method for Determining Reaction Pathways and Locating Molecular Transition States. *Chem. Phys. Lett.* **1977**, *49*, 225−232. (b) Behn, A.; Zimmerman, P. M.; Bell, A. T.; Head Gordon, M. Incorporating Linear Synchronous Transit Interpolation into the Growing String Method: Algorithm and Applications. *J. Chem. Theory Comput.* **2011**, *7*, 4019−4025.
(63) Bell, S.; Crighton, J. S. Locating Transition States. *J. Chem. Phys.* **1984**, *80*, 2464−2475.
(64) Czerminski, R.; Elber, R. Reaction Path Study of Conforma tional Transitions in Flexible Systems: Applications to Peptides. *J. Chem. Phys.* **1990**, *92*, 5580−5601.
(65) Moré, J. J.; Garbow, B. S.; Hillstrom, K. E. *User Guide for MINPACK 1*; Report ANL 80 74; Argonne National Laboratory: Argonne, IL, 1980.
(66) Kirkpatrick, S.; Gelatt, C. D.; Vecchi, M. P. Optimization by Simulated Annealing. *Science* **1983**, *220*, 671−680.
(67) (a) Griewank, A. O. Generalized Descent for Global Optimization. *J. Optimiz. Theory Appl.* **1981**, *34*, 11−39. (b) Butler, R. A. R.; Slaminka, E. E. An Evaluation of the Sniffer Global Optimization Algorithm Using Standard Test Functions. *J. Comput. Phys.* **1992**, *99*, 28−32.
(68) Kolossváry, I.; Guida, W. C. Low Mode Conformational Search Elucidated: Application to $C_{39}H_{80}$ and Flexible Docking of 9 Deazaguanine Inhibitors into PNP. *J. Comput. Chem.* **1999**, *20*, 1671−1684.
(69) (a) Li, Z.; Scheraga, H. A. Monte Carlo Minimization Approach to the Multiple Minima Problem in Protein Folding. *Proc. Natl. Acad. Sci. U. S. A.* **1987**, *84*, 6611−6615. (b) Wales, D. J.; Doye, J. P. K. Global Optimization by Basin Hopping and the Lowest Energy Structures of Lennard Jones Clusters Containing up to 110 Atoms. *J. Phys. Chem. A* **1997**, *101*, 5111−5116.

DOI: 10.1021/acs.jctc.8b00529
J. Chem. Theory Comput. 2018, 14, 5273−5289

(70) (a) Kostrowicki, J.; Scheraga, H. A. Application of the Diffusion Equation Method for Global Optimization to Oligopeptides. *J. Phys. Chem.* **1992**, *96*, 7442−7449. (b) Nakamura, S.; Hirose, H.; Ikeguchi, M.; Doi, J. Conformational Energy Minimization Using a Two Stage Method. *J. Phys. Chem.* **1995**, *99*, 8374−8378. (c) Pappu, R. V.; Hart, R. K.; Ponder, J. W. Analysis and Application of Potential Energy Smoothing for Global Optimization. *J. Phys. Chem. B* **1998**, *102*, 9725−9742. (d) Pappu, R. V.; Marshall, G. R.; Ponder, J. W. A Potential Smoothing Algorithm Accurately Predicts Transmembrane Helix Packing. *Nat. Struct. Biol.* **1999**, *6*, 50−55.

(71) Ma, J.; Straub, J. E. Simulated Annealing Using the Classical Density Distribution. *J. Chem. Phys.* **1994**, *101*, 533−541.

(72) Rogers, J. W., Jr.; Donnelly, R. A. Potential Transformation Methods for Large Scale Global Optimization. *SIAM J. Optim.* **1995**, *5*, 871−891.

(73) (a) Beeman, D. Some Multistep Methods for Use in Molecular Dynamics Calculations. *J. Comput. Phys.* **1976**, *20*, 130−139. (b) Brooks, B. R. *Algorithms for Molecular Dynamics at Constant Temperature and Pressure*; DCRT Report; National Institutes of Health: Bethesda, MD, 1988.

(74) (a) Lelièvre, T.; Rousset, M.; Stoltz, G. Langevin Dynamics with Constraints and Computation of Free Energy Differences. *Math. Comput.* **2012**, *81*, 2071−2125. (b) Lelièvre, T.; Stoltz, G.; Rousset, M. *Free Energy Computations: A Mathematical Perspective*; Imperial College Press: London, 2010.

(75) Martyna, G. J.; Tuckerman, M. E.; Tobias, D. J.; Klein, M. L. Explicit Reversible Integrators for Extended Systems Dynamics. *Mol. Phys.* **1996**, *87*, 1117−1157.

(76) Bussi, G.; Zykova Timan, T.; Parrinello, M. Isothermal Isobaric Molecular Dynamics Using Stochastic Velocity Rescaling. *J. Chem. Phys.* **2009**, *130*, 074101.

(77) (a) Qian, X.; Schlick, T. Efficient Multiple Time Step Integrators with Distance based Force Splitting for Particle Mesh Ewald Molecular Dynamics Simulations. *J. Chem. Phys.* **2002**, *116*, 5971−5983. (b) Humphreys, D. D.; Friesner, R. A.; Berne, B. J. A Multiple Time Step Molecular Dynamics Algorithm for Macro molecules. *J. Phys. Chem.* **1994**, *98*, 6885−6892.

(78) Smith, W. Hail Euler and Farewell: Rotational Motion in the Laboratory Frame. *CCP5 Newsletter*, February 2005.

(79) Andersen, H. C. Rattle: A "Velocity" Version of the Shake Algorithm for Molecular Dynamics Calculations. *J. Comput. Phys.* **1983**, *52*, 24−34.

(80) Allen, M. P. Brownian Dynamics Simulation of a Chemical Reaction in Solution. *Mol. Phys.* **1980**, *40*, 1073−1087.

(81) Guarnieri, F.; Still, W. C. A Rapidly Convergent Simulation Method: Mixed Monte Carlo/Stochastic Dynamics. *J. Comput. Chem.* **1994**, *15*, 1302−1310.

(82) Shi, Y. y.; Lu, W.; van Gunsteren, W. F. On the Approximation of Solvent Effects on the Conformation and Dynaics of Cyclosporin A by Stochastic Dynamics Simulation Techniques. *Mol. Simul.* **1988**, *1*, 369−383.

(83) Bussi, G.; Donadio, D.; Parrinello, M. Canonical Sampling through Velocity Rescaling. *J. Chem. Phys.* **2007**, *126*, 014101.

(84) Berendsen, H. J. C.; Postma, J. P. M.; van Gunsteren, W. F.; DiNola, A.; Haak, J. R. Molecular Dynamics with Coupling to an External Bath. *J. Chem. Phys.* **1984**, *81*, 3684−3690.

(85) Andersen, H. C. Molecular Dynamics Simulations at Constant Pressure and/or Temperature. *J. Chem. Phys.* **1980**, *72*, 2384−2393.

(86) Evans, D. J.; Holian, B. L. The Nose−Hoover Thermostat. *J. Chem. Phys.* **1985**, *83*, 4069−4074.

(87) Frenkel, D.; Smit, B. *Understanding Molecular Simulation: From Algorithms to Applications*, 2nd ed.; Academic Press: New York, 2001.

(88) (a) Leimkuhler, B.; Margul, D. T.; Tuckerman, M. E. Stochastic, Resonance Free Multiple Time Step Algorithm for Molecular Dynamics with Very Large Time Steps. *Mol. Phys.* **2013**, *111*, 3579−3594. (b) Minary, P.; Martyna, G. J.; Tuckerman, M. E. Algorithms and Novel Applications based on the Isokinetic Ensemble. I. Biophysical and Path Integral Molecular Dynamics. *J. Chem. Phys.* **2003**, *118*, 2510−2526.

(89) Kaledin, A. L.; Kaledin, M.; Bowman, J. M. All Atom Calculation of the Normal Modes of Bacteriorhodopsin Using a Sliding Block Iterative Diagonalization Method. *J. Chem. Theory Comput.* **2006**, *2*, 166−174.

(90) Zwanzig, R. W. High Temperature Equation of State by a Perturbation Method. I. Nonpolar Gases. *J. Chem. Phys.* **1954**, *22*, 1420−1426.

(91) Bennett, C. H. Efficient Estimation of Free Energy Differences from Monte Carlo Data. *J. Comput. Phys.* **1976**, *22*, 245−268.

(92) Daly, K. B.; Benziger, J. B.; Debenedetti, P. G.; Panagiotopoulos, A. Z. Massively Parallel Chemical Potential Calculation on Graphics Processing Units. *Comput. Phys. Commun.* **2012**, *183*, 2054−2062.

(93) Wyczalkowski, M. A.; Vitalis, A.; Pappu, R. V. New Estimators for Calculating Solvation Entropy and Enthalpy and Comparative Assessments of their Accuracy and Precision. *J. Phys. Chem. B* **2010**, *114*, 8166−8180.

(94) Bell, D. R.; Qi, R.; Jing, Z.; Xiang, J. Y.; Mejias, C.; Schnieders, M. J.; Ponder, J. W.; Ren, P. Calculating Binding Free Energies fo Host Guest Systems Using the AMOEBA Polarizable Force Field. *Phys. Chem. Chem. Phys.* **2016**, *18*, 30261−30269.

(95) Stone, A. J. Distributed Multipole Analysis: Stability for Large Basis Sets. *J. Chem. Theory Comput.* **2005**, *1*, 1128−1132.

(96) Wu, J. C.; Chattree, G.; Ren, P. Automation of AMOEBA Polarizable Force Field Parameterization for Small Molecules. *Theor. Chem. Acc.* **2012**, *131*, 1138.

(97) Sullivan, F.; Mountain, R. D.; O'Connellf, J. Molecular Dynamics on Vector Computers. *J. Comput. Phys.* **1985**, *61*, 138−153.

(98) Sagui, C.; Pedersen, L. G.; Darden, T. A. Towards an Accurate Representation of Electrostatics in Classical Force Fields: Efficient Implementation of Multipolar Interactions in Biomolecular Simu lations. *J. Chem. Phys.* **2004**, *120*, 73−87.

(99) Frigo, M.; Johnson, S. G. The Design and Implementation of FFTW3. *Proc. IEEE* **2005**, *93*, 216−231.

(100) Wang, W.; Skeel, R. D. Fast Evaluation of Polarizable Forces. *J. Chem. Phys.* **2005**, *123*, 164107.

(101) Simmonett, A. C.; Pickard, F. C., IV; Shao, Y.; Cheatham, T. E., III; Brooks, B. R. Efficient Treatment of Induced Dipoles. *J. Chem. Phys.* **2015**, *143*, 074115. (b) Simmonett, A. C.; Pickard, F. C., IV; Ponder, J. W.; Brooks, B. R. An Empirical Extrapolation Scheme for Efficient Treatment of Induced Dipoles. *J. Chem. Phys.* **2016**, *145*, 164101.

(102) 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.* **2015**, *143*, 174104.

(103) Albaugh, A.; Niklasson, A. M. N.; Head Gordon, T. Accurate Classical Polarization Solution with No Self Consistent Field Iterations. *J. Phys. Chem. Lett.* **2017**, *8* (8), 1714−1723.

(104) 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.* **2017**, *147*, 161724.

(105) (a) Zheng, L.; Chen, M.; Yang, W. Random Walk in Orthogonal Space to Achieve Efficient Free Energy Simulation of Complex Systems. *Proc. Natl. Acad. Sci. U. S. A.* **2008**, *105*, 20227−20232. (b) 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.* **2009**, *130*, 234105.

(106) Abella, J. R.; Cheng, S. Y.; Wang, Q.; Yang, W.; Ren, P. Hydration Free Energy from Orthogonal Space Random Walk and Polarizable Force Field. *J. Chem. Theory Comput.* **2014**, *10*, 2792−2801.

(107) Schnieders, M. J.; 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. *J. Chem. Theory Comput.* **2012**, *8*, 1721−1736.

DOI: 10.1021/acs.jctc.8b00529
J. Chem. Theory Comput. 2018, 14, 5273−5289

(108) Zheng, L.; Yang, W. Practically Efficient and Robust Free Energy Calculations: Double Integration Orthogonal Space Tempering. *J. Chem. Theory Comput.* **2012**, *8*, 810−823.
(109) Crippen, G. M.; Havel, T. F. *Distance Geometry and Molecular Conformation*; Research Studies Press, Ltd.: Somerset, England, 1988.
(110) Mucherino, A.; Lavor, C.; Liberti, L.; Maculan, N. *Distance Geometry: Theory, Methods and Applications*; Springer: New York, 2013.
(111) Wuthrich, K. *NMR of Proteins and Nucleic Acids*; Wiley Interscience: New York, 1986.
(112) Kuszewski, J.; Nilges, M.; Brunger, A. T. Sampling and Efficiency of Metric Matrix Distance Geometry: A Novel Partial Metrization Algorithm. *J. Biomol. NMR* **1992**, *2*, 33−56.
(113) Dionne, R. Étude et Extension d'un Algorithme de Murchland. *INFOR* **1978**, *16*, 132−146.
(114) Oshiro, C. M.; Thomason, J.; Kuntz, I. D. Effects of Limited Input Distance Constraints Upon the Distance Geometry Algorithm. *Biopolymers* **1991**, *31*, 1049−1064.
(115) Hodsdon, M. E.; Ponder, J. W.; Cistola, D. P. The NMR Solution Structure of Intestinal Fatty Acid Binding Protein Complexed with Palmitate: Application of a Novel Distance Geometry Algorithm. *J. Mol. Biol.* **1996**, *264*, 585−602.
(116) Huang, E. S.; Samudrala, R.; Ponder, J. W. Distance Geometry Generates Native like Folds for Small Helical Proteins Using Consensus Distances of Predicted Protein Structures. *Protein Sci.* **1998**, *7*, 1998−2003.
(117) Kim, S.; Thiessen, P. A.; Bolton, E. E.; Chen, J.; Fu, G.; Gindulyte, A.; Han, L.; He, J.; He, S.; Shoemaker, B. A.; Wang, J.; Yu, B.; Zhang, J.; Bryant, S. H. PubChem Substance and Compound Databases. *Nucleic Acids Res.* **2016**, *44*, D1202−13.
(118) Berman, H. M.; Westbrook, J.; Feng, Z.; Gilliland, G.; Bhat, T. N.; Weissig, H.; Shindyalov, I. N.; Bourne, P. E. The Protein Data Bank. *Nucleic Acids Res.* **2000**, *28*, 235−242.
(119) Yin, J.; Henriksen, N. M.; Slochower, D. R.; Shirts, M. R.; Chiu, M. W.; Mobley, D. L.; Gilson, M. K. Overview of the SAMPL5 Host−Guest Challenge: Are We Doing Better? *J. Comput. Aided Mol. Des.* **2017**, *31*, 1−19.
(120) Feenstra, K. A.; Hess, B.; Berendsen, H. J. C. Improving Efficiency of Large Time Scale Molecular Dynamics Simulations of Hydrogen Rich Systems. *J. Comput. Chem.* **1999**, *20*, 786−796.
(121) Gresh, N. Development, Validation, and Applications of Anisotropic Polarizable Molecular MechanicsTto Study Ligand and Drug−Receptor Interactions. *Curr. Pharm. Des.* **2006**, *12*, 2121−2158.
(122) Cisneros, G. A.; Piquemal, J. P.; Darden, T. A. Generalization of the Gaussian Electrostatic Model: Extension to Arbitrary Angular Momentum, Distributed Multipoles, and Speedup with Reciprocal Space Methods. *J. Chem. Phys.* **2006**, *125*, 184101.
(123) (a) Rackers, J. A.; Wang, Q.; Liu, C.; Piquemal, J. P.; Ren, P.; Ponder, J. W. An Optimized Charge Penetration Model for Use with the AMOEBA Force Field. *Phys. Chem. Chem. Phys.* **2017**, *19*, 276−291. (b) Narth, C.; Lagardère, L.; Polack, E.; Gresh, N.; Wang, Q.; Bell, D. R.; Rackers, J. A.; Ponder, J. W.; Ren, P. Y.; 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.* **2016**, *37*, 494−506. (c) Wang, Q.; Rackers, J. A.; He, C.; Qi, R.; Narth, C.; Lagardere, L.; Gresh, N.; Ponder, J. W.; Piquemal, J. P.; Ren, P. General Model for Treating Short Range Electrostatic Penetration in a Molecular Mechanics Force Dield. *J. Chem. Theory Comput.* **2015**, *11*, 2609−2618.
(124) (a) 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.* **2014**, *10*, 1638−1651. (b) 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.* **2015**, *11*, 2589−2599.

DOI: 10.1021/acs.jctc.8b00529
J. Chem. Theory Comput. 2018, 14, 5273−5289