πDMD User's Guide

Version 1.0

Program Authors: Feng Ding, David G. Shirvanyants, Shuangye Yin, Nikolay V. Dokholyan
Documentation: Elizabeth A. Proctor

November 15, 2011

Molecules in Action, LLC
1530 Mill Valley Road
Chapel Hill, NC 27516

The πDMD User's Guide describes the performance of molecular simulations using the parallel simulation engine πDMD. The guide includes details of the algorithm, force field, and molecular models, as well as descriptions of input and output files and the options of the πDMD suite.

Copyright

Non-commercial Use

Software Registration

License Agreement

Contact Information

For licensing issues, email:

license@moleculesinaction.com

Molecules in Action, LLC

1530 Mill Valley Road

Chapel Hill, NC 27516

For software support, email:

support@moleculesinaction.com

Table of Contents

1 Introduction

πDMD is a parallelized molecular dynamics simulation engine optimized for the simulation of macromolecular complexes at biologically relevant time and length scales. This User's Guide will give some background on the technique and describe how to use the πDMD suite.

1.1 Discrete Molecular Dynamics Simulations

The simulation method Discrete Molecular Dynamics (DMD) computes particle trajectories by applying the principles of physics. In place of the continuous potentials used in traditional molecular dynamics (MD) simulations, DMD employs discrete step function potentials. Bonded interactions (bonds, bond angles, and dihedrals) are modeled as infinite square wells, and non-bonded interactions are represented as a series of discrete energetic steps, decreasing in magnitude with increasing distance until reaching zero at some cutoff distance.

As a result, the simulation engine solves the ballistic equations of motion for only those particles participating in a collision, instead of integrating over Newton's equations of motion for every particle in the system. Each particle in the simulation has a constant velocity until reaching the interaction distance of another particle (termed a "collision"), upon which it instantaneously changes velocity based on the laws of conservation of energy, momentum, and angular momentum. Each time step represents a single collision of particles, and the time between time steps is therefore variable; valuable calculation time is not spent on system snapshots where no events are occuring. To this end, following each time step DMD utilizes a rapid sorting algorithm to identify the next collision in the system. Because fewer calculations are performed, the DMD method allows for longer time and length scales to become accessible in the simulation of large biomolecules. A full discussion of the DMD algorithm can be found in (1-3).

1.2 πDMD – Parallel Discrete Molecular Dynamics

The parallel implementation of DMD is considered to be intrinsically difficult. The reason for this difficulty is that DMD simulations are event-driven; every subsequent event is computed from the current atom positions and velocities, which themselves result from a preceding chain of events. DMD events include atom collisions and non-collision events needed to model

thermostat, hydrogen bonding, and to keep track of nearest neighbors. Any two events in DMD are potentially coupled; that is, the outcome of a preceding atomic collision may affect the time and place of the subsequent events. Thus, predicting many collisions in parallel is problematic, since after the first collision other predictions may become invalid. However, the coupling of collisions is limited in time and space. When a collision occurs between atoms *i* and *j*, the effect propagates through the system with a finite average speed. Therefore, many of the earlier collision predictions will remain valid if the atoms participating in those predictions are located sufficiently far from both *i* and *j* that the effect of the *i-j* collision does not have time to propagate to their location before their collision.

The event-based parallelization approach utilized in πDMD splits the DMD simulation cycle into several stages (4). First, every collision event in the system is predicted based on the current atom positions and velocities. Using the predicted collision time, πDMD computes new atom coordinates and velocities. However, unlike the serial DMD algorithm, in πDMD the states of the atoms are not immediately updated. Instead, the results of the collision outcome are stored at a temporary memory location. Every event is then tested for supersession by an earlier collision; such events are cancelled and their temporarily stored outcomes are discarded. Finally, events that have not been cancelled are "committed," and the results previously stored in temporary memory are copied to the primary storage of atom properties. Collision prediction, evaluation, and exclusion can be performed simultaneously for most events, while the commitment stage is executed only serially. In a typical DMD simulation, event prediction is the most computationally intensive component, thus its parallelization produces the largest performance gain.

Thread synchronization is the most important step in πDMD simulations, distinguishing πDMD from the serial DMD algorithm. Two or more threads must never simultaneously modify the same shared data; the result of such unsynchronized data access is unpredictable. Synchronization is usually performed by the introduction of a "lock mechanism," which allows one thread to access the data while all other threads wait. Coupled events must also be detected, in order to ensure that they are processed in a serial manner. Thread locking and coupled events lead to wasted CPU cycles, with adverse effects for parallelization efficiency. The performance of thread synchronization strongly affects the overall πDMD performance, as every collision evaluation requires at least one synchronization point employing a lock mechanism, which may cause threads to waste time waiting for one another. In order to minimize the locking overhead, the πDMD algorithm uses only non-blocking locks.

1.3 Examples of Molecular Models

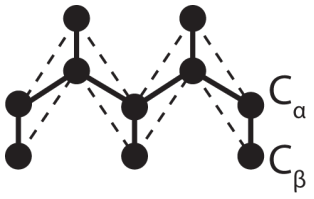
πDMD has the ability to utilize multi-scale modeling techniques in order to accommodate varying molecular systems. Depending on the complexity, flexibility, and desired resolution scale of the system of interest, molecules may be represented atom-by-atom or at a coarse-grained level. In biological systems, simplified models are often desirable in that they speed up the simulation process and allow for access to greater time- and length-scales. However, when a greater level of detail is needed, an all-atom model is the best choice.

1.3.1 1-bead Protein Model

In the one-bead protein model, each residue is represented as a single sphere, forming a 'beads-on-a-string' polymer model of the protein. The πDMD potentials utilize the structure-based Gō model; residues in contact in the native structure attract each other, and those that are not in contact repulse each other. A matrix of contacts is assigned so that, for residues *i* and *j*, if the distance between the residues in the native structure r_{ij} is such that the attractive spheres of the residues overlap, the element matrix ε_{ij} has a value of 1. If r_{ij} is greater than this maximum interaction distance, then ε_{ij} is assigned as −1. The contact matrix is then used to calculate pairwise interaction potentials, and the potential energy of the structure is a sum of these pairwise interactions, $E = \frac{1}{2}\sum_{i,j=1}^{N} U_{ij}$.

The one-bead protein model is useful for long time- and length-scale protein folding studies, such as the study of protein aggregation. This model has been shown to admirably capture the thermodynamic and kinetic properties of folding dynamics (2).

1.3.2 2-bead Protein Model

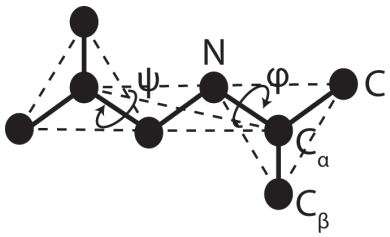


In the two-bead protein model, each residue is represented by one sphere for the C_α atom and one sphere for the C_β atom, with the exception of glycine, which has only the

C_α atom. The introduction of an additional bead per residue limits the backbone flexibility of the protein and allows for the modeling of high folding cooperativity (5).

The two-bead protein model is useful for the characterization of protein transition states. The transition-state ensemble is the focus of many protein design efforts because of the ability of a protein in this collection of conformational states to change rapidly from folded to unfolded forms.

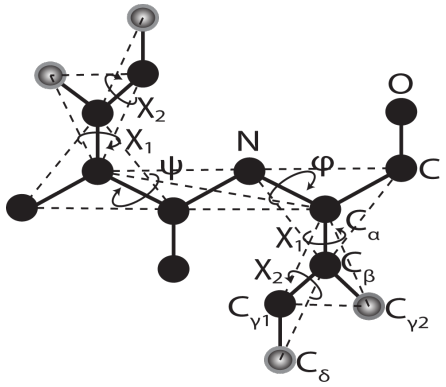
1.3.3 4-bead Protein Model



In the four-bead protein model, each residue is represented by three backbone spheres (C, C_α, and N) and one sphere for the C_β atom, with the exception of glycine, which has no C_β atom. The additional backbone atoms allow for the modeling of N_i–C_j hydrogen bonds. The intermediate resolution of the four-bead protein model introduces additional calculation over the two-bead model, due to the additional beads used in the model and the necessary hydrogen-bonding interactions in the backbone. One method to compensate for these extra calculations is to introduce experimentally-determined constraints to the protein system in order to decrease the accessible conformational space (6). A set of inter-residue proximity constraints reduces the number of degrees of freedom of the system, and therefore the number of possible collisions and amount of necessary calculations.

The four-bead protein model is useful in the study of secondary structure transitions (7). Representation of all backbone atoms results in the ability to distinguish secondary structure formation and transitions in secondary structure, due to the accurate modeling of backbone–backbone hydrogen bonds.

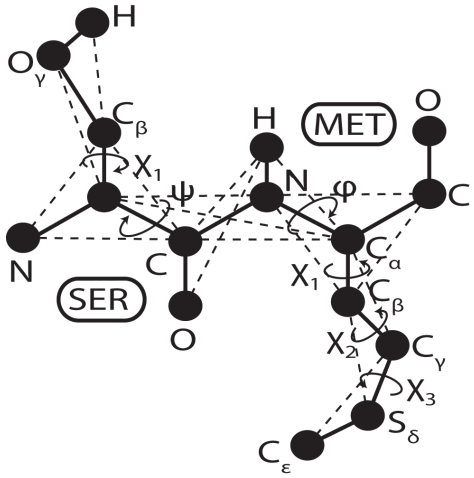
1.3.4 Quasi-all-atom Protein Model



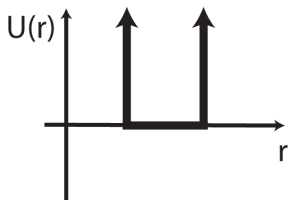
In the quasi-all-atom protein model, each residue is represented as in the four-bead model, with the addition of a C_γ atom to all residues except glycine and alanine. The beta-branched residues, threonine, isoleucine, and valine, have a second C_γ atom to represent the second brach after the C_β atom. The bulky residues, arginine, lysine, and tryptophan, are assigned an additional C_δ atom to simulate their additional volume.

The quasi-all-atom model is useful for modeling protein core packing and side-chain degrees of freedom (8). The quasi-all-atom regime improves on the four-bead model in that it can account for side-chain entropy and the effect of side-chain size on core packing.

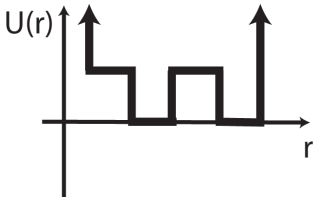
1.3.5 All-atom Protein and RNA Models



In the all-atom model, all heavy atoms and polar hydrogen atoms are explicitly represented.



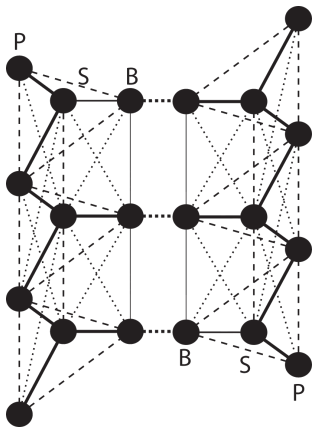
Bonded interactions are modeled using infinite square well potentials. For covalently bonded atoms *i* and *j*, the potential U_{ij} is equal to $+\infty$ for r_{ij} outside the range of the bond length ± the bond variance. This effectively renders covalent bonds as unbreakable, and two covalently bound atoms will be permanently constrained by this interaction for the duration of the simulation. Angular constraints on next-nearest-neighbor atoms *i* and *j*, with $j = i + 2$, are similarly represented by an infinite square well potential, which is dependent on that of the covalent bond constraints for consecutive atoms.



Dihedral interactions between atoms *i* and *j*, with $j = i + 3$, are modeled by a multistep potential function of pairwise distance r_{ij}. The set of distance parameters (d_{min}, d_0, d_1, d_2, and d_{max}) for these potentials is experimentally determined from distance distributions in a non-redundant database of high-resolution protein structures.

The all-atom protein model of DMD is useful for the study of conformational dynamics on the atomic level (3). Studies such as protein design, drug screening, and protein-protein and protein-RNA interactions benefit from the high resolution of the all-atom model.

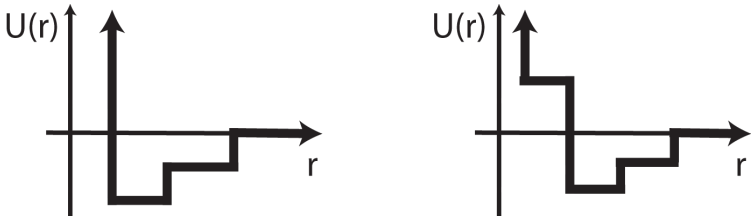
1.3.6 Coarse-grained RNA and DNA Models



Because of the increased degrees of freedom in ribonucleic acid (RNA) as compared with proteins, RNA has a much greater flexibility. The conformational space of RNA increases exponentially with length, making a coarse-grained model of RNA invaluable to conformational sampling. RNA is modeled as three 'beads-on-a-string,' with beads representing the sugar and phosphate groups positioned at the respective centers of mass of each group, and the nucleo-base bead positioned at the geometric center of the hexagonal base ring.

In a similar way to RNA, deoxyribonucleic acid (DNA) in complex with histone can be represented using a 'beads-on-a-string' coarse-grained model for DNA. The beads are the same as those used in the RNA model described above: one bead for sugar, located at the centroid of the deoxyribose ring; one bead for phosphate, located at the centroid of that group's constituent atoms; and one bead for the base, located at the centroid of the hexagonal ring.

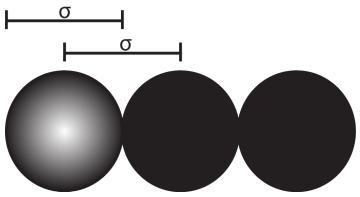
For both nucleic acids, neighboring beads are constrained by bond length, bond angle, and dihedrals. The parameters for these bonded interactions are calculated from high-resolution known RNA or DNA structures. Non-bonded interactions important for the determination of secondary and tertiary structure include base pairing, base stacking, phosphate–phosphate repulsion, and hydrophobic interactions. Parameters for these interactions are experimentally determined. Bonded interactions, consisting of covalent bonds, bond angles, and dihedrals, are modeled as infinite square-well potentials, as in the protein models. Base-stacking constraints are also modeled by an infinitely high potential well, while sugar-base steric constraints are defined by hard-sphere repulsion.



Base-pairing interactions are modeled as two-step potential functions (left). Interactions between DNA and histones consist of hydrogen bonding between the basic residues lysine and arginine of the histone and the acidic phosphates in DNA, modeled as three-step potential functions (right). Because of the negative charge of the DNA molecule, interactions between DNA and the acidic or nonpolar residues of histone are negligible, and are not included in the model.

In combination with the coarse-grained nature of the RNA and DNA models, the rapid conformational sampling provided by the πDMD algorithm allows for the efficient exploration of conformational space, despite the flexibility of these molecules (9,10). The number of degrees of freedom in the RNA and DNA systems can be further limited using experimental constraints, such as sets of pairwise inter-nucleotide distances (11). These techniques used in concert sufficiently simplify the systems to a degree where computational simulation is feasible and practical.

1.3.7 Coarse-grained Lipid Model



The three-bead lipid model is useful for the study the self-organization of lipids (12). The first bead represents both the lipid head group and the glycerol backbone, and the second and third beads each represent a fatty acid chain. All beads are of identical diameter, σ. Both bonding and flexibility potentials are represented as discrete square-well functions. Flexibility is governed by the angle formed between the two bonds in the three-bead chain. The bond angle potential is also represented as an infinite square well as a function of distance between the first bead and the last bead.

Intermolecular attractive interactions between fatty acid chains are modeled by a two-step discrete potential. For distances between σ and σ + w, where w is an adjustable parameter, the potential is −ε. The tail–tail potential is set to zero for distances greater than σ + w, and infinity for less than the hard-sphere radius. Head-head and head-tail intermolecular pairs are defined as non-interacting, with the exception of hard-sphere repulsion.

1.3.8 Non-biological systems

πDMD is not only a simulation engine for biological molecules, but in fact has its roots in non-biological chemical and physical systems (13). Any system for which a PDB formatted file exists or is created may be explored with πDMD simulations. The πDMD package contains parameters for most atom types, including some metal ions. πDMD is ideal for systems with large degrees of freedom, for example the simulation of liquids and artificial materials.

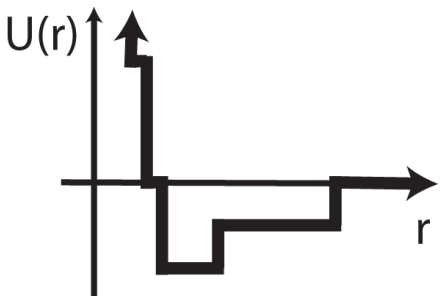
1.4 Medusa Force Field

All-atom πDMD utilizes the Medusa force field, a weighted sum of seven energetic terms:

$$E = W_{vdw\_attr}E_{vdw\_attr} + W_{vdw\_rep}E_{vdw\_rep} + W_{solv}E_{solv} + W_{es}E_{es} + W_{bb\_hbond}E_{bb\_hbond} + W_{sc\_hbond}E_{sc\_hbond} + W_{bb\_sc\_hbond}E_{bb\_sc\_hbond}$$

where E_{vdw_attr} and E_{vdw_rep} are the attractive and repulsive parts of the van der Waals (VDW) interaction, E_{solv} is the solvation energy, E_{es} is the electrostatic interaction potential, and E_{bb_hbond}, E_{sc_hbond}, and $E_{bb\_sc\_hbond}$ are the hydrogen bonding energies for backbone-backbone, side chain-side chain, and backbone-side chain interactions, respectively. A full description of the Medusa force field can be found in (14-16).

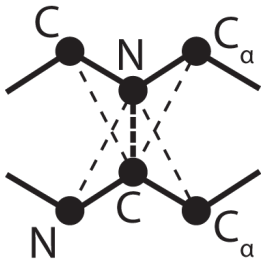
The VDW interaction model and parameters for the Medusa force field are adapted from CHARMM19 (17). Medusa utilizes the EEF1 implicit solvent model proposed by Lazaridis and Karplus (18) and the hydrogen bonding model proposed by Kortemme and Baker (19). The cutoff distance for all non-bonded interactions is 9.0 Å.



The Medusa force field is discretized for use in all-atom πDMD by mimicking the attractions and repulsions between interacting pairs. The VDW and solvation energies are modeled as pairwise functions of distance. Both the Lennard-Jones and Lazaridis-Karplus functions are fit to a multi-step square-well function, characterized by the atomic hardcore radius and a series of potential steps. The potential steps are based on pairwise distances, the set of which parameters are determined from experimentally-determined distributions in a non-redundant database of high resolution protein structures.

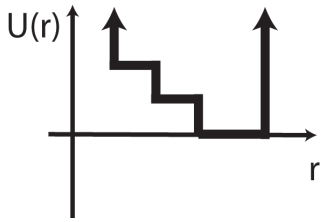
The Medusa force field includes electrostatic interactions between charged residues, including basic and acidic residues. The central atoms of charged groups (CZ for Arg, NZ for Lys, CG for Asp, and CD for Glu) are assigned integer charges. Screened charge-charge interactions are modeled by the Debye-Hückel approximation. The Debye length is set at 10 Å, assuming a monovalent electrolyte concentration of 0.1 mM. The water relative permittivity is set to 80 for computation of the screened charge-charge interaction potential. The continuous electrostatic interaction potential is discretized for use in πDMD with an interaction range of 30 Å; beyond this range, the screened potential is zero.

1.5 Hydrogen Bonding Reaction Algorithm



Hydrogen bonding reactions are modeled using the Reaction Algorithm. Upon the formation of a hydrogen bond between non-adjacent residues *i* and *j*, N_i and C_j change their atom types to N'_i and C'_j, respectively. Whether or not a hydrogen bond will form depends not only on the

interacting atoms, but also on interactions between the neighbors of these atoms, called auxiliary interactions. Distances between neighboring atoms are included in the calculations in order to mimic the orientation-dependence of the hydrogen bond interaction. Once N and C reach the appropriate distance range for interaction, the auxiliary interaction distances are evaluated in order to assign the total potential energy change between N_i–C_j and other surrounding atoms both before and after the putative hydrogen bond formation.



A two-step potential is used when assigning energy to auxiliary interactions, in order to allow some angular distortion in exchange for an energetic penalty. If the auxiliary interaction distances satisfy the pre-determined range, and the total kinetic energy of the interacting atoms (minus any angular distortion penalty) is enough to overcome the potential energy change, formation of the hydrogen bond is allowed. If these conditions are not satisfied, the hydrogen bond does not form. All possible interactions between backbone–backbone, backbone–side chain, and side chain–side chain atoms are included. A full discussion of the Reaction Algorithm can be found in (7).

2 πDMD with Replica Exchange

Replica exchange is a technique developed to enhance the exploration of the potential energy landscape during molecular simulations. This enhanced sampling aids in the complete description of the structure and thermodynamics of molecular systems, the central theme of most molecular modeling applications.

At any given temperature, the ruggedness of the free energy landscape and the slope toward local minima govern the structural sampling efficiency. Simulation at high temperatures can accelerate escape from local minima and therefore improve sampling, but the free energy landscape is then altered by the larger entropic contributions brought about by thermal structural fluctuations. In order to overcome energy barriers while still maintaining energetically-relevant conformational sampling, the replica exchange method utilizes multiple simulations (replicas) of

the same system in parallel, with each simulation performed at a different temperature (20,21). The temperature range used usually corresponds to a spectrum of structures, from native-like to melted. At periodic time intervals, simulation temperatures are exchanged between replicas based on a Monte Carlo-type function. Temperatures are exchanged between two replicas *i* and *j*, maintained at temperatures T_i and T_j and with energies E_i and E_j, according to the canonical Metropolis criterion with the exchange probability *p*:

$$p = \min\left(1,\ e^{(E_i - E_j)\left(\frac{1}{k_B T_i} - \frac{1}{k_B T_j}\right)}\right)$$

πDMD utilizes the Anderson thermostat to maintain constant temperature during simulation (22).

3 Input Files

Input to πDMD comprises two main parts: structural information and energetic information. Structural information consists of the atom types, coordinates, velocities, and connectivity of the system. Energetic information consists of the bonded and non-bonded potentials that will be applied to the system, as well as system temperature and heat exchange parameters. In addition, πDMD requires instructions for the simulation engine, such as how for how long to run the simulation and how often to write the output files. The files in which this information is supplied are described below.

3.1 File Formats

πDMD requires at least three input files, and up to five, depending on the type of simulation and the desired constraints or long-range potentials. In this section, we will describe the formats of these input files and the information that they contain.

3.1.1 Parameter File

The parameter file contains both structural and energetic information.

NUMBER OF ATOMS

The total number of atoms in the system, including particles of infinite mass (>10^8 Da) that are used as “anchor points” for implementing system constraints.

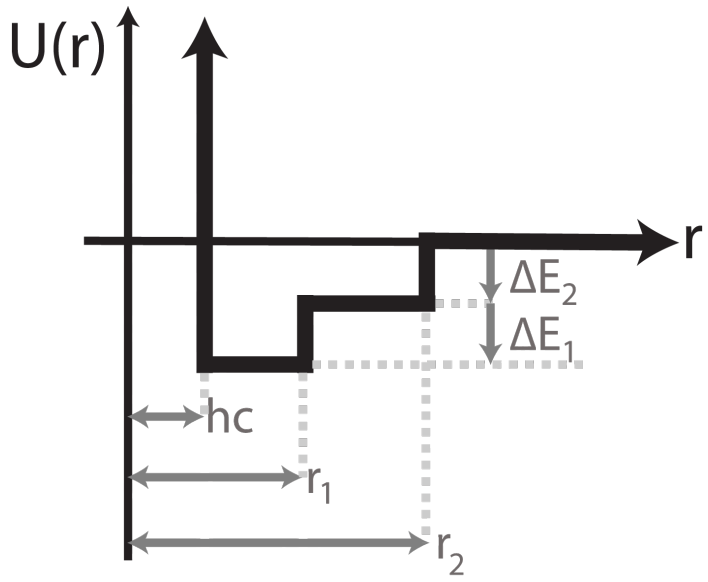
ATOM TYPES

A list of the atom types with each type given a sequential number starting with 1. Each atom type is followed by its respective mass and radius. For example:

```
#type mass        radius
  1   12.000000 1.000000
  2   12.000000 1.055000
  3   12.000000 1.500000
  4   12.000000 1.590000
  5   12.000000 1.620000
```

NON-BONDED POTENTIALS

A list of each atom type interacting with each other atom type, and the respective non-bonded potential steps for their interactions. The two atom types are listed, followed by the hard-core radius, and then the radii and energies of the successive potential steps. For example:



#type1	type2	hc	r_1	ΔE_1	r_2	ΔE_2
1	1	0.1000	0.2900	-1.0000	0.4800	-1.0000

If the non-bonded interaction potential is not defined, πDMD will treat their interaction as a simple hard sphere collision, where the interaction radius is the sum of the two corresponding hard-core radii, given in ATOM TYPES.

REACTION POTENTIALS

Reaction potentials describe the potential steps for all possible second-order chemical reactions between the various atom types that involve the formation of a bond. The resulting bond may be permanent, in which case the last potential step will be the limit of an infinite square well and the atoms are not allowed to move outside of reaction range, or non-permanent, in which case it will be broken if the atoms move outside the distance of the last potential step. Whether the bond is permanent or non-permanent, all second-order reactions are reversible. The probability of breaking the bond formed in a second-order reaction is determined by reaction energy, not by the shape of the reaction potential. The reaction potentials are defined using the same format as the un-bonded potentials, above, with the exception of the last energetic term being removed if the reaction results in a permanent bond. Removal of this term results in an infinite square well limit by default:

#type1	type2	hardcore	Δr_1	ΔE_1	Δr_2	ΔE_2	Δr_3	ΔE_3	Δr_3	…
89	91	1.76	2.20	-0.60	2.50	-0.60				
12	91	2.64	3.10	-0.60	3.50	-10.00	3.99	10.60	4.00	

REACTIONS

Second-order chemical reactions between various atom types are described in the reactions section. When two atoms undergo a second-order chemical reaction, they change their atom types. The format of this section lists the old atom types, the new atom types after the reaction, whether the reaction results in the formation of a bond (1 or 0), the length of the bond formed (or the interaction distance, if the reaction forms a non-permanent bond), the change in energy, and the difference in reactor index necessary for the interaction to occur. The difference in reactor index is assigned in order to avoid forming second-order bonds within the same residue or between residues adjacent in sequence. The reactor index is described in the REACTION LIST, below.

#old1	old2	new1	new2	isbond	d	dE	delta_idx
1	15	89	90	1	2.5	0	4

REACTION LIST

The reaction list identifies the atoms that are capable of undergoing second-order chemical reactions, such as hydrogen bonds. If the atom index is not listed in this section, that atom will never be considered for forming a hydrogen bond. Each atom is given a reactor index, which places it in a group of atoms that should not form hydrogen bonds between themselves. Atoms with the same reactor index, or with a difference in reactor index less than that defined for the reaction (see REACTIONS, above), will not undergo a second-order chemical reaction. Also listed for each atom are the neighboring atoms that will undergo auxiliary interactions upon the formation of a hydrogen bond (see section 1.5 on the hydrogen bonding reaction algorithm). An example entry on the reaction list:

#atomIndex	reactorIndex	reactorAssociates
5	1	4
11	2	10
20	3	19
21	3	19

In this example, atom 5 has been assigned to reactor index group 1, and atom 4 will undergo an auxiliary interaction when atom 5 forms a hydrogen bond. Atoms 20 and 21 have both been assigned to reactor index group 3, meaning that, with a reactor index difference of 0, the reaction listed above (REACTIONS), with a required difference in reactor index of 4, will never form between atoms 20 and 21. When either atom 20 or 21 undergoes a second-order reaction, atom 19 will undergo an auxiliary interaction.

BONDED POTENTIALS

Bonded potentials describe the step potentials for various types of covalent bonds. The format of this section includes a potential index, which is referenced later in the connectivities section, below, and the potential that describes that specific covalent bond type. The potentials described here have the same format as in the unbounded potentials, with the exception that the last step does not have an energetic value attached; this final step is the limit of the infinite square well that describes bonded interactions in πDMD. In this implementation of bonded interactions, a covalent bond may never be broken.

CONNECTIVITIES

The connectivities section describes which atoms are covalently bonded, and the specific type of covalent bond by which they are linked. The format for this section lists the two atom indices and the potential index (described in BONDED POTENTIALS).

3.1.2 State File

The state file contains a snapshot of the system. This file contains the coordinates and velocities of each atom in the system, as well as the indices of the chain and residue to which the atom belongs, in the format: atom index, atom type, x, y, z, v_x, v_y, v_z, chain index, residue index. The chain index and residue index values are optional. The state file also contains a line giving the dimensions of the simulation box. For example, the first few lines of a state file:

```
DIMENSION
   300.00000     300.00000     300.00000
ATOMS
    1   26  147.08  145.65  137.17   -1.61   -0.31    0.95 1 1
    2   24  148.30  146.06  137.61   -0.95    0.25    1.70 1 1
    3    3  146.01  146.63  137.40   -0.85   -0.31    0.70 1 1
```

3.1.3 Constraint File

The constraint file contains the pairwise long-range potentials generated from user-specified constraints. These constraints are given in the format of the two atom indices, followed by the potential steps. The potential steps are described as in the non-bonded and bonded potentials in the parameter file, above.

3.1.4 Start File

The start file contains the various control parameters for the system, such as the simulation temperature, heat exchange coefficient, simulation time, and the filenames and write times for the output files. For example, a short single-temperature simulation:

```
T_NEW           0.55
T_LIMIT         0.55
HEAT_X_C        0.1
RESTART_FILE    dmd_restart
```

```
RESTART_DT          1000
ECHO_FILE           dmd_echo
ECHO_DT             10
MOVIE_FILE          dmd_movie
MOVIE_DT            100
START_TIME          0
MAX_TIME            100000
```

Section 5.1 gives a full explanation of all control parameters in this file.

3.1.5 Replica Exchange Input File

In order to perform replica exchange simulations using πDMD, a replica exchange input file should be specified in the command line with the flag "-r" (details of command line parameters and options can be found in section 5.2). This input file will contain information on the number of replicas desired, their desired temperatures in reduced units (T (K) • k_B (kcal/mol•K)), and their restart files (when applicable). This input file will also contain a name and file path for the replica exchange output file, described below. For example, a replica exchange file for 4 replicas:

```
N_REPLICA      4
RX_DT          1000

REPLICA_STATE 0      p000.dmd_restart
REPLICA_STATE 1      p001.dmd_restart
REPLICA_STATE 2      p002.dmd_restart
REPLICA_STATE 3      p003.dmd_restart

REPLICA_TEMP  0      0.50
REPLICA_TEMP  1      0.55
REPLICA_TEMP  2      0.60
REPLICA_TEMP  3      0.65

RX_OUT         RX_TEMP.out
```

Section 5.2 gives a full explanation of all control parameters in this file.

3.2 Simulations With Constraints

In addition to the coarse-grained potentials and parallelization found in πDMD, the introduction of experimental results as system constraints is an effective method to compensate for the many calculations involved in the simulation of large biological molecules. The number of degrees of freedom of the system is reduced by the incorporation of constraints, which decreases the size of accessible conformational space, and therefore the number of possible collisions and amount of calculation necessary to describe the dynamics of the system.

3.2.1 Creating Input Constraints

During the simulation setup process, the user can create a set of constraints to apply to the system in order to decrease the degrees of freedom and, with it, simulation time. These constraints can come in several formats:

AtomPair AtomSelect#1 AtomSelect#2 Potential

Here, "AtomPair" is a keyword that denotes to the program that what follows is a pairwise potential. "AtomSelect#1" is the first atom interacting in the constraint potential, defined by the chain index, residue index, and atom name in the PDB file, separated by ".". The atom definitions are followed by the potential steps, as discussed above in section 3.1.1 describing the parameter file. For example:

```
AtomPair  1.111.CA  2.1.SG2 2.0 3.364 1.2 4.094 -1.2  7.5
```

Static AtomSelections

The "Static" keyword denotes to the program that the following atoms should maintain fixed position during simulation. The designated atoms are held fixed by anchoring them to particles of infinite mass (>10^8 Da). "AtomSelections" defined in the same manner as "AtomSelect," above, with the additional ability to denote multiple atoms, residues, and chains by using "," to separate distinct entities, "-" to separate two integers in order to define a set of integers, and "*" as a wildcard to declare all of the chains, residues, or atoms. For example:

```
Static 1-2.*.N,CA
```

This line denotes that the backbone nitrogen and α-carbon atoms of all residues in chains 1 and 2 should be held static.

Static 1-2,4.12,19.*

This line denotes that all atoms of residues 12 and 19 in chains 1, 2, and 4 should be held static.

Static 1.1-4.CA

This line denotes that all α-carbon atoms of residues 1 to 4 in chain 1 should be held static.

Harmonic AtomSelections

The "Harmonic" keyword denotes to the program that a harmonic potential should be applied to constrain the following atoms within a certain distance of their initial coordinates. The AtomSelections are given in the same manner as for the "Static" keyword, above.

3.2.2 Incorporating Additional Potentials

In addition to individual pairwise constraints, long-range potentials may also be implemented in the constraints file. The pairwise nature of constraints also lends itself to the expression of pairwise potentials. In order to incorporate long-range potentials into πDMD simulations, the potential needs only to be expressed in a pairwise manner and written in the input constraints file with the AtomPair keyword for each pair.

3.3 Generating πDMD Input Files

In all-atom πDMD, the main files directly input to the simulation engine can be generated by a conversion program, complex.linux. This program reads the PDB and other structural files, as well as user constraints, if applicable, and combines the information contained therein with the Medusa force field parameters (section 1.4) to generate input files that the program can more easily read. The exception is the start file, which contains basic simulation parameters such as simulation temperature and length, and output filenames.

3.3.1 Parameter, State, and Constraint Files

The parameter, state, and constraint files that will be input to the simulation engine are generated from structural files supplied by the user by the program complex.linux:

Usage:

complex.linux -P paramDir -I complexPDB –D dimension –p outParam –s outstate

paramDir

This argument gives the path to the Medusa parameter directory. The program uses parameters from the Medusa force field in order to interpret the structure and output the necessary atom interactions and pairwise potentials.

complexPDB

This argument gives the path of the PDB file containing the coordinates that will be used to initiate the simulations.

dimension

This argument gives the dimensions of the simulation box as “x,y,z” with no spaces in between. If a single value is given, the simulation box will be cubic.

outParam

This argument gives the name of the output parameter file, which will be an input for the πDMD simulations.

outState

This argument gives the name of the output state file, which will be an input for the πDMD simulations.

In addition to these mandatory arguments, further optional arguments may be given in order to address simulation specifics:

-T newTopParamFile

This argument gives the path to the topology-parameter file, which contains the filenames and paths to MOL2 files used for small molecule, ion, or lipid ligands. The format of this file includes the “MOL” keyword, followed by the PDB name of the molecule and the path to the file. For example:

```
MOL GSH   ./GSH.mol2
```

New atom types may also be defined in the topology-parameter file, using the keyword “ATOM TYPE,” followed by the name of the atom type and its non-bonded (VDW and solvation) interaction potential.

-S seed

This argument gives the seed for the random number generator. In order to perform reproducible simulations, this parameter should be set to a non-zero integer value. If this parameter is not specified, the default random seed is -111.

-C inConst

This argument gives the path to the user-generated input constraints file, described above in section 3.2.

-c outConst

This argument gives the name of the output constraint file, which will be an input for the πDMD simulations. Please note that this file is different than the user-generated input constraint file for complex.linux.

-Z dzLip

This argument is used in the simulation of membrane proteins, and gives the thickness of the coarse-grained lipid membrane.

-n

Usage of this argument will avoid the translation of the PDB coordinates. By default, the system coordinates are translated so that the center of mass of the system is located at the origin in the simulation box.

-d

Usage of this argument will idealize the bonds and angles of the system, minimizing and removing clashes according to the parameters of the Medusa force field.

3.3.2 Start File

The start file contains the control parameters for the simulation. These control parameters include simulation start and end times, simulation temperature, and the names of the πDMD output files and how often they are written. Definitions for these control parameters are contained in section 5, and an example start file is provided in section 3.1.5.

3.4 Creating Input Files for Coarse-grained Models

The πDMD engine is able to simulate the dynamics of a broad range of molecular systems, given the proper input files. As discussed above, the parameter and constraint files provide the types and potentials for particle interactions, while the state file contains the coordinates and velocities of those particles. In this section, we will first discuss the general principles of defining a molecular system and creating the parameter and constraint files. Then, we will outline examples of various coarse-grained developed for DMD simulations, discussed in greater detail in Section 1.3.

In order to physically model the dynamics of a molecular system, we need to define particles and assign their interactions. These particles may consist of atoms in an all-atom model, or may represent groups of atoms or residues in a coarse-grained model. For simplicity, we will simply refer to particles as “atoms” for the duration of this User’s Guide; any occurrence of “atom” can also be taken to mean “particle.” The particles are defined using atom types, and the assigned interactions include both the non-bonded and bonded potentials between the various atom types.

First, one needs to identify all atom types in the system, and determine their corresponding masses and hardcore radii. By default, each atom pair is assigned only the non-bonded interaction of hardcore (perfectly elastic) collision, where the hardcore distance is the sum of the hardcore radii of the two atoms. Additional interactions can be defined as non-bonded potentials (NON-BONDED POTENTIALS, Section 3.1.1). We note that non-bonded interactions are defined in a pairwise manner, according to the interacting atom types.

Given that the majority of biomolecules are polymers, we need to introduce bonded interactions in order to model their shape and geometry. Bonded interactions include covalent bond (atoms *i* and *i+1*), bond angles (atoms *i* and *i+2*), and dihedrals (atoms *i* and *i+3*). A definition of bonded interaction potentials can be derived from statistical analysis of PDB structures. The format for describing bonded interactions and atom connectivity is discussed above in Section 3.1.1. We note that the connectivity of bonded interactions is defined by atom indices.

In all but the very coarsest-grain molecular models, hydrogen bonding interactions must be included for proper simulation of dynamics. Hydrogen bonding in πDMD is modeled by the reaction algorithm (Section 1.5). The concept of the reaction algorithm is that an atom of type A and an atom of type B can change their types to type A' and B', respectively, when the two atoms come within a given interaction range. Therefore, the interactions associated with these new atoms types, including both non-bonded and bonded interactions, must be defined. The non-bonded interactions related to the reacted atoms are treated identically to other non-bonded interactions, previously discussed. The bonded interactions associated with "reacted" atoms, however, are treated differently. The bonded reaction interactions include 1) a temporary bond assigned between two reacted atoms; and 2) auxiliary bonds (see Section 1.5) that define the angular dependence of the temporary bond. These reaction-related bond interactions are defined in the reaction potentials section (REACTION POTENTIALS, Section 3.1.1), and the reactions themselves are defined in the reactions section (REACTIONS, Section 3.1.1). The reaction list section (REACTION LIST, Section 3.1.1) defines the reactor associates (the auxiliary atoms for each reaction-competent atom) and the reactor index that is used to prevent reactions between neighboring reactants along a polymer sequence.

The constraints file (described in detail in Section 3.1.3) can be used to define additional interactions that are not defined in the parameter file. By default, interactions described in the constraint file will override other interactions that the atoms otherwise would have had with each other, although this behavior can be changed using the -fb flag when initiating πDMD simulations (Section 5.1). In πDMD simulations, the interactions between two atoms are assigned only one of the possible types of interactions, with the following priorities: 1) constraints, 2) bonded potentials, and 3) reaction-associated (temporary) bonds, 4) non-bonded potentials. In this way, the constraint file provides an additional flexibility to the modeling of molecular systems in πDMD.

In summary, the input parameter and constraint files for πDMD provide flexibility in the definition of molecular systems. As a result, the user can define any arbitrary model to mimic the system of interest. Next, we will describe as examples several coarse-grained models that have been previously developed. We will focus mainly on the definition of atoms types and bonded and non-bonded interactions.

3.4.1 1-bead Protein Model

The one-bead model, described in Section 1.3.1, uses only the α-carbon atom, C_α, to represent each amino acid. Bonded interactions include the peptide bond between neighboring C_α atoms ($C_{\alpha i}$, $C_{\alpha i+1}$), and the next nearest neighboring C_α atoms ($C_{\alpha i}$, $C_{\alpha i+2}$). The average distance between $C_{\alpha i}$ and $C_{\alpha i+1}$ is approximately 3.83 Å. The standard deviation of these distances can be derived from statistical analysis of PDB structures. Because the bond length variation determines the frequency of bond oscillation, which consists of the largest portion of CPU time in πDMD, a wider bond variation may be used, given the condition that wider bond variation does not significantly distort the structure. In this case, the bond variation is defined as ±2% of the average C_α-C_α distance. The potential steps for the infinite square well of the bond are therefore defined as:

```
3.75    3.91
```

For the bonded interaction between ($C_{\alpha i}$, $C_{\alpha i+2}$), statistical analysis of the distance distributions suggests two peaks, corresponding to alpha-helical (peaks at 5.30 Å and 5.60 Å) and extended (peaks at 6.25 Å and 7.20 Å) conformations:

```
5.30  5.60  -E_barrier  6.25 7.20
```

Here, E_barrier is the energy barrier, which can be defined according to the energy scales.

For non-bonded interactions, the structure-based Gō model (23,24) may be used to ensure that the lowest-energy state is native-like. In the Gō model, the number of atom types should be proportional to the number of atoms, since the interaction matrix is N×N, where N is the number of residues. Native contacts are assigned attractive interactions:

```
3.0  8.0  -E_attr
```

where “3.0” corresponds to the hardcore distance, “8.0” corresponds to the maximum interaction range, and E_attr is the attractive energy. Non-native contacts have weaker repulsive interactions:

3.0 8.0 E_rep

where the E_rep is a repulsive energy. The value of E_rep can be negative, but with |E_rep| < |E_attr| (a weaker attraction as compared to native contacts), or positive (repulsion). In the case where E_rep is zero, the non-native interactions are reduced to:

3.0

Two approaches may be used to define the non-bonded interactions:

1) Define N atom types. In the non-bonded potentials section (NON-BONDED POTENTIALS, Section 3.1.1), we define interaction potentials for each pair of atom types, depending on whether they are native or non-native contacts.
2) Define one atom type. In the non-bonded poentials section (NON-BONDED POTENTIALS, Section 3.1.1), we define only the non-native interactions. In the constraint files, we define the native contacts with “capped” interaction potentials, which produce an infinite square well:

   3.0 8.0 -E_attr D_MAX

Where D_MAX can be defined as a very large value, so that the two atoms cannot exceed this distance while in the simulation box. Here, native contacts are defined according to their atom indices.

By carefully choosing the energy values of E, E_attr, and E_rep, a 1-bead protein model may be built. For example, E_attr = E_rep = 1.0 kcal/mol, and E_barrier = 2.0kcal/mol.

3.4.2 2-bead Protein Model

In the two-bead protein model, each amino acid is represented by two atoms, the α-carbon (C_α) and β-carbon (C_β), as described in Section 1.3.2. The bonded interactions include $C_{\alpha i}$-$C_{\beta i}$, $C_{\alpha i}$-$C_{\beta i+1}$, $C_{\beta i}$-$C_{\alpha i+1}$. $C_{\alpha i}$-$C_{\alpha i+1}$, and $C_{\alpha i}$-$C_{\alpha i+2}$. Here, the first three terms (those related to C_β) are in addition to the 1-bead protein model. Based on statistical

analysis, as well as practical considerations, we define the interaction potential of $C_{\alpha i}$-$C_{\beta i}$ as:

1.49 1.57

$C_{\alpha i}$-$C_{\beta i+1}$ as:

4.65 4.77 E_barrier 4.90

and $C_{\beta i}$-$C_{\alpha i+1}$ as:

4.40 4.80

Here, E_barrier is the energy that determines the probability of crossing the barrier.

Non-bonded interactions may be assigned according the Gō model (23,24) in order to ensure that the ground state is native-like. For simplicity, we choose the C_β atom (C_α atom for glycine) as the interaction center for each amino acid. In the C_β-based contacts, 7.5 Å is a commonly-used interaction range. Other non-bonded interactions are represented as hardcore collisions. Similar to the 1-bead protein model, we may adopt two approaches to define the molecular system:

1) Define N+1 atom types, which include the C_α type along with N Gō-interaction centers. In the non-bonded potentials section (NON-BONDED POTENTIALS, Section 3.1.1), all pairwise interaction potentials are defined, depending on whether the atom pair is native, non-native, or will be modeled by a hardcore collision.
2) Defined 2 atom types. In the non-bonded potentials section of the parameter file (NON-BONDED POENTIALS, Section 3.1.1), only the non-native interactions are described. In the constraint file, native contacts with “capped” interaction potentials are defined, as described for the 1-bead model.

3.4.3 4-bead Protein Model

In the four-bead protein model, the N, C, C_α, and C_β atoms are represented, as described in Section 1.3.3. In addition, the atoms N and C may form hydrogen bonds and change their type to N’ and C’. The bonded interactions are defined in detail in (7). In the case of Gō-interaction models, the same approaches may be used as in the one- and two-bead models. We can also assign hydrophobic-hydrophilic (HP)-like potentials

(7). Here, we will focus only on the reaction modeling, which occupies three sections: REACTION POTENTIALS, REACTIONS, and REACTION LIST.

REACTIONS

A description of the reactions section can be found in Section 3.1.1. In the four-bead protein model, reactions are modeled as the following:

```
N  C  N'  C'  1  4.2 0  4
```

Here, N, C, N', and C' are the atom type indices, 1 suggests that a temporary bond is assigned between bonded atoms, 4.2 is the maximum reaction range, and 4 is the cutoff of difference in reactor indexes, smaller than which two reactors will not react. This cutoff is used to exclude calculation of reactor collisions along the protein backbone that cannot form hydrogen bonds. Since the reaction range is 4.2 Å, a potential step should exist in the non-bonded interaction potential between N and C with a distance of 4.2 Å. The energy corresponding to this step can be zero. For example:

```
N  C  3.0  4.20  0.00
```

Although the described potential is essentially a hardcore collision, the step should be defined in order for the πDMD engine to check whether a reaction is feasible upon colliision.

REACTION LIST

A description of the reaction list section can be found in Section 3.1.1. In the four-bead protein model, the reaction list is produced as the following:

```
1_C  1  1_CA  2_N
2_N  2  1_C  2_CA
2_C  2  2_CA 3_N
3_N  3  2_C  3_CA
….
```

Here, 1_C is the atom index of carton atom of residue 1. Similarly, 1_N is the atom index of N atom of residue 1, and 1_CA is the α-carbon of residue 1. The second column uses the residue index as the reactor index. The third and fourth columns define the associated auxiliary atoms used to define hydrogen bonds.

REACTION POTENTIALS

A description of the reaction potentials section can be found in Section 3.1.1. In the four-bead protein model, reaction potentials are modeled as the following:

```
N'  C'   4.00 4.20 –E_hb
N'  CA  4.46  4.66  E_hb/2  4.82  E_hb/2  5.56
N'  N    4.47  4.62  E_hb/2  4.78  E_hb/2  5.41
N'  N'   4.47  4.62  E_hb/2  4.78  E_hb/2  5.41
C'  C    4.40  4.56  E_hb/2  4.72  E_hb/2  5.39
C'  C'   4.40  4.56  E_hb/2  4.72  E_hb/2  5.39
C'  CA  4.44  4.62  E_hb/2  4.79  E_hb/2  5.39
```

Combining the above sections, hydrogen bonding interactions may be effectively modeled the reaction algorithm.

3.4.4 Quasi-all-atom Protein Model

In the quasi-all-atom model, additional side-chain beads and the carbonyl oxygen in the backbone supplement the atoms included in the four-bead model, as described in Section 1.3.4. Specifically, a gamma-bead (C_γ) is included for cysteine (Cys), methionine (Met), phenylalanine (Phe), leucine (Leu), tyrosine (Tyr), serine (Ser), glutamine (Gln), asparagine (Asn), glutamic acid (Glu), aspartic acid (Asp), and histidine (His) residues. For the beta-branched residues such as valine (Val), isoleucine (Ile), and threonine (Thr), we include the a second gamma bead, $C_{\gamma 2}$. For the bulky residues tryptophan (Trp), arginine (Arg), and lysine (Lys), we include an additional delta bead (C_δ). Detailed interaction parameters can be found in the (8), including parameters for both bonded and non-bonded interactions.

In addition to backbone-backbone hydrogen bonds, the quasi-all-atom model also includes side-chain-backbone hydrogen bonding interactions. Specifically, the hydrogen bond acceptors include the $C_{\gamma 2}$ bead of threonine, and the C_γ bead of serine, asparagine, and aspartic acid. The hydrogen bond donors include the $C_{\gamma 2}$ bead of threonine, and the C_γ bead of asparagine, glutamine, and serine. Interaction details can be found in the supporting material of (8). By following the principles described in Section 3.4.3, both

side-chain-backbone and backbone-backbone hydrogen bonds can be effectively modeled.

3.4.5 Coarse-grained RNA and DNA Models

In the coarse-grained RNA or DNA model, each nucleotide is represented by the phosphate (P), sugar (S), and base (B) bead, as described in Section 1.3.6. The P bead corresponds to the phosphate atom, the S bead represents the center of mass of the 5-atom sugar pucker, and the B bead corresponds to the centroid of the six-atom ring in both purine and pyrimidine bases.

The details of the interaction parameters for coarse-grained DNA molecules are described in (5). Base pairs are constrained without the explicit modeling of hydrogen bonds. Therefore, definition of the bonded and non-bonded interaction potentials in the parameter and constraint files is straightforward.

In the case of RNA, the goal of simulations is often to predict the folding of a single-stranded RNA into specific structures. Base pairs are modeled by hydrogen bonds between complementary nucleotides. Details of the interaction parameters for base-pairing interactions can be found in (9). We note that, due to strong coupling between hydrogen bonding and stacking interactions, simulation of coarse-grained RNA may not be parallelized.

3.4.6 Coarse-grained Lipid Model

The specifics of the coarse-grained lipid model are discussed in Section 1.3.7. The implementation of the coarse-grained lipid model in πDMD is very straightforward, without the complication of hydrogen bonding interactions. Following the same principles described above, and applying the interaction parameters defined in (12), one may create the input parameter and constraint files for molecular modeling of the coarse-lipid system.

4.3 Restart File

At an interval specified by the user in the start file (see section 5.1 for how to specify this interval), πDMD will save a snapshot of the system particle locations and velocities. This snapshot, saved in the restart file, can serve as a "bookmark" of sorts for restarting the simulation if an error or run-time limit should occur. The restart file has the same format as the input state file (described in detail in section 3.1.2), and upon restarting the simulation, will be given in place of the state file after the "-s" flag (see sections 5.1 and 5.2 for command line usage, or πDMD Tutorial section 2.4 for an example of restarting a πDMD simulation). The restart file is overwritten at each interval, so that at any given time there exists only one snapshot, the most recent "bookmark."

4.4 Replica Exchange Output File

The πDMD protocol outputs a file that tracks the temperature of each replica throughout the simulation. At a user-specified interval during the simulation, one line prints the time step followed by each temperature. For instance, in a simulation with 4 replicas and a RX_DT of 2000:

```
0.0000     0.5000     0.5500     0.6000     0.6500
2000.0000  0.5000     0.6000     0.5500     0.6500
4000.0000  0.5500     0.6500     0.5000     0.6000
6000.0000  0.5000     0.6500     0.5500     0.6000
```

This information is necessary for the calculation of replica exchange rate, which is an indicator of proper replica spacing and sampling. The πDMD package includes a Perl script, rx_rate.pl, for calculating the exchange rate from the replica exchange output file.

5 Program Options and Simulation Parameters

4 Output Files

πDMD produces three output files in single-trajectory mode, and four for replica exchange simulations. In this section, we will describe the file formats and the type of information that they contain.

4.1 Movie (Trajectory) File

The primary output file of πDMD is the movie file, or the trajectory. The movie file tracks the coordinates of each atom at a user-specified interval, given in the start file (see section 5.1 for how to specify this parameter). The movie file is in binary format, so as to economize disk space. In order to convert the movie file to a human-readable format, the πDMD package provides the complex_M2P program, the usage of which is outlined in section 7.

4.2 Echo File

The πDMD echo file tracks the energetic and thermodynamic information of the system as the simulation progresses. At an interval specified by the user in the start file (see section 5.1 for how to specify this parameter), πDMD will report the time step, average temperature, average pressure, average potential free energy, instantaneous potential free energy, and the instantaneous kinetic energy and write these parameters to the echo file. For example, the first 50 time steps of a simulation with ECHO_DT specified as 10:

# Time	aveTemperature	avePressure	avePotential	instPotential	instKineticE
10.000	0.44242	0.0000000017	-268.12470	-269.74085	192.05882
20.000	0.45152	0.0000001196	-259.78069	-265.06802	196.31415
30.000	0.45984	0.0000000184	-262.38298	-263.20668	204.69417
40.000	0.46091	-0.0000001110	-264.70496	-261.02619	191.51733
50.001	0.44770	0.0000000762	-268.62775	-272.51062	187.09288

Time is measured in πDMD time steps, temperature values are given in πDMD reduced units (T (K) • k_B (kcal/mol•K)), pressure values are given in kcal/mol•Å^3, and energetic values are given in kcal/mol.

πDMD is capable of performing both single trajectory and replica exchange simulations. Single trajectory simulations are useful for studying native-state dynamics. Replica exchange simulations provide enhanced sampling by performing multiple simulations in parallel at a range of temperatures, and periodically swapping the system temperatures. In this way, efficiency is increased by allowing the system to overcome free energy barriers while maintaining conformational sampling according to the relevant free energy surface.

5.1 Single Trajectory Simulations

Usage:

pdmd.linux [OPTIONS] -i start_file -p param_file -s state_file

The parameter, state, and start files are required inputs to πDMD. The generation of these files has been discussed in Section 3. In brief, the parameter file contains parameters for atom types and bonded and non-bonded potentials, the state file contains a system snapshot of particle positions and velocities from which to start the simulation, and the start file contains the simulation control parameters. These parameters include:

HEAT_X_C

This parameter denotes the heat exchange coefficient. The heat exchange coefficient indicates the rate of heat transfer between the thermostat-maintained implicit solvent and the system. Roughly, dt = 1/HEAT_X_C. This rate will need to be adjusted depending on simulation temperature, temperature change over the simulation, and the total simulation time. For single-temperature trajectories, a common heat exchange coefficient is 0.1.

T_NEW

This parameter denotes the starting temperature of the simulation, in reduced units: T (K) • k_B (kcal/mol•K). If this parameter is specified, the velocities contained in the input state file will be re-scaled accordingly.

T_LIMIT

This parameter denotes the ending temperature of the simulation, the maximum (or minimum) temperature that will be attained.

RESTART_FILE

This parameter specifies the name of the output restart file. The default name of this file is "dmd_restart." The restart file is overwritten during the simulation in order to provide only the most current “bookmarked” snapshot of the system.

ECHO_FILE

This parameter specifies the name of the output file for the energy, pressure, and temperature at given time steps (see ECHO_DT, below)

MOVIE_FILE

This parameter specifies the name of the movie (trajectory) output file.

RESTART_DT

This parameter specifies the number of time steps between saving the restart file. The default value of this parameter is 1000.

ECHO_DT

This parameter specifies the number of time steps between saving the energy, pressure, and temperature of the system.

MOVIE_DT

This parameter specifies the number of time steps between writing the snapshot of the system to the movie file.

MOVIE_SAVE_START

This parameter specifies the starting atom to be saved. The default of this parameter is 1. The value of this parameter should NOT be changed without extensive knowledge of the system.

MOVIE_SAVE_END

This paramter specifies the last atom in the system to be saved. The default value of this parameter is the last atom in the system according to the input state file. The value of this parameter should NOT be changed without extensive knowledge of the system.

START_TIME

This parameter specifies the initial time when the simulation starts. The default value of this parameter is 0. If a value other than 0 is given, the output files will be appended instead of overwritten.

MAX_TIME
This parameter specifies the maximum number of time steps for the simulation. At the given time, the simulation will be complete.

RANDOMSEED
This parameter controls the random seed for the random number generator. In order to perform reproducible simulations, this parameter should be set to a non-zero integer value. If this parameter is not specified, the system time will be used as the random seed.

COMPRESS
This parameter will compress the output files in gzip format when set to an integer value 1-9. Compression typically will reduce the size of the movie file by half. Alternatively, any output file name (see above) can be specified with a ".gz" extension in order to compress only this file. Compressed (gzip) input is recognized automatically, and does not depend on file extension.

For advanced tuning of the πDMD algorithm, higher-level users may refer to the source documentation for further control parameters. Additional options for πDMD can be controlled using various command line flags:

-c <path>
This flag denotes the location of a user-created file that contains additional constraints or long-range potentials to amend the force field.

-m <number>
This flag enables multithreading using <number> logical CPUs.

-fa
This flag controls the overwriting of πDMD output files. When this flag is used, the output files will be appended. Without this flag, the files will be overwritten by default, or appended if a start time other than 0 is given.

-fb

This flag will allow the use of both the generic force field bonded potentials and the user-created long-range constraints (when the “-c” flag is used). Without this flag, the user-created constraints will replace the bonded interactions for those atom pairs included in the user-created constraint file.

-fn

This flag will allow the use of both the generic force field non-bonded potentials and the user-created long range constraints (when the “-c” flag is used). Without this flag, the user-created constraints will replace the non-bonded interactions for those atom pairs included in the user-created constraint file.

-fl

Use of this flag will treat those atom pairs interacting via additional long-range potentials as covalently bonded when writing the output movie file. When viewing the trajectory using molecular visualization software, these atoms wil appear as covalently bonded.

-fc

Use of this flag moves the center of mass of the system to the center of simulation box at start.

-fh

Use of this flag permits atoms to form multiple hydrogen bonds, as opposed to only one by default.

-fr

Use of this flag disables rounding to 0. If this flag is not used, the lower bounds of all potentials that are equal to or less than the threshold (currently 1e-05) will be rounded to 0.0 by default.

-ft

Use of this flag disables the detection of hyper-threading. Without this flag, the default behavior is to detect and avoid hyper-threading.

-fx

Use of this flag enables output of an extended movie format with 32-bit integers and floats. Without this flag, the default output movie format is 16-bit.

-q

Quiet option. Use of this flag suppresses most program messages. This flag can be specified up to three times in order to suppress minimal program output and warnings. Critical errors that terminate the program are not suppressed.

-v

Verbose option. Use of this flag increases program verbosity; more messages will print to the command line.

5.2 Replica Exchange Simulations

Usage:

rexpdmd.linux [OPTIONS] -i start_file -p param_file -s state_file -r rx_file

Replica exchange simulations employ the exact same parameters and option as do single trajectory simulations, with the addition of a replica exchange file. The replica exchange file is optional in that the parameters included in this file may instead be stated in the start file. However, in the case where both single trajectory and replica exchange simulations are being performed for the same system, or in the interest of maintaining generic start files, these parameters may instead be included in a separate replica exchange file, denoted here by the “-r” flag. The control parameters used in this file include:

N_REPLICA

This parameter denotes the number of replicas that will be employed in replica exchange simulations.

RX_DT

This parameter denotes the number of time steps between attempted swaps of temperature replicas. This parameter is also the spacing of each line of output in the replica exchange output file, which records the temperatures of each replica at a given time.

REPLICA_STATE

This parameter is only relevant to simulations that are being restarted. This parameter should be followed by the sequential number of the replica, starting with zero, and the name of the desired restart file. For instance, in a system with 4 replicas:

```
REPLICA_STATE 0     p000.dmd_restart
REPLICA_STATE 1     p001.dmd_restart
REPLICA_STATE 2     p002.dmd_restart
REPLICA_STATE 3     p003.dmd_restart
```

When initiating the simulation for the first time, this parameter is not included. Without specification, the state file used for each replica will be the same, as indicated in the command line with the "-s" flag.

REPLICA_TEMP

This parameter denotes the simulation temperature for each replica. This parameter should be followed by the sequential number of the replica, starting with zero, and the simulation temperature for each replica, in reduced units: T (K) • k_B (kcal/mol•K). For instance, in a system with 4 replicas:

```
REPLICA_TEMP  0     0.56
REPLICA_TEMP  1     0.60
REPLICA_TEMP  2     0.63
REPLICA_TEMP  3     0.66
```

Relevant temperatures will vary by system. These temperatures will override the temperature denoted in the start file with the T_NEW or T_LIMIT control parameters.

RX_OUT

This parameter denotes the name of the output replica exchange file, which records the temperatures of each replica at a given time.

6 Running πDMD

The speed-up of πDMD over the DMD serial algorithm is most dramatic when running on large computing clusters with many nodes available. However, πDMD is fast and efficient enough to perform meaningful simulations on a laptop computer.

6.1 System Architecture

πDMD may be used on single workstations with multiple cores, networks of single workstations, or computing clusters. πDMD requires multiple cores for parallel scaling, but will also run on single-core machines using the non-parallelized DMD algorithm. The software does not require parallel extensions in order to compile. When utilizing networks of workstations for πDMD simulation, it is advised that the machines to be homogenous. For replica exchange simulations, especially, efficiency will be limited by the speed of the slowest node. Every RX_DT time steps (see Section 5.2 for a description of this control parameter), the separate simulation replicas will attempt to swap temperatures, and will need to wait for all replicas to reach the same simulation time. If nodes in the network are of significantly different speeds, the simulation will effectively pause to wait for the slowest node to catch up to the others in simulation time before attempting the exchange, potentially losing any benefit in speed-up.

6.2 Memory Usage and Scaling

The high rate of data exchange between threads causes πDMD to be highly dependent on the speed of memory access. On modern processors, such the Intel Xeon or AMD Opteron, the highest exchange rate is achieved between the cores of a single multi-core CPU. Therefore, the best scaling of πDMD is achieved when all threads run on CPU cores on the same dye.

πDMD performance highly depends upon the average number of neighbors per atom. Generally, simulations of compact objects, such as collapsed globular proteins, are more computationally costly than simulations of dilute systems such as unfolded proteins. Compact objects have on average more buried atoms, which have a greater number of neighbors than do surface atoms, and require more calculations to predict the next collision.

In large systems, a randomly chosen pair of atoms is likely to have a greater distance between them than in small systems. Large systems therefore enjoy a decrease in the probability of event coupling, and hence a decrease in the fraction of cancelled events (wasted calculations).

In πDMD simulations of large biological molecules, the decrease in event coupling compensates for the increase in number of calculations, resulting in a nearly linear dependence of simulation time on protein length (Shirvanyants et al. (2011), *submitted*).

7 Analysis of πDMD Simulations

πDMD trajectories are output in binary format. Analysis programs may either be designed to directly read the movie file format, or the PDB format conversion program (included in the πDMD package) may be used to output a human-readable format:

Usage:

complex_M2P.linux <paramDir> <complexPDB> <newTopParamList> <movie> <outPDB> [constraints [startFrame [nFrames [dFrames]]]]

paramDir

This argument gives the path to the Medusa parameter directory. The program uses parameters from the Medusa force field in order to interpret the trajectory and output the PDB file format.

complexPDB

This argument gives the path of the PDB file that was used to create the simulation input files. The PDB file is used to reassign the identities of each atom in the trajectory snapshots.

newTopParamList

This argument gives the path to the topology-parameter file, which contains the filenames and paths to MOL2 files used for small molecule or ion ligands. If a topology-parameter file was not used, the path /dev/null can be used for this argument instead.

movie

This argument gives the path to the movie file that will be converted to PDB format.

outPDB

This argument gives the path and filename of the desired output PDB file. The output PDB file will contain all specified snapshots of the trajectory, each separated by the “ENDMDL” card.

constraints
This optional argument gives the simulation input constraints, if used.

startFrame
This optional argument gives the desired start frame for PDB conversion. The default value is 1 (starting from the beginning of the simulation). If a number less than 1 is given, the program will revert to default behavior. In order to specify this argument, the “constraints” argument (above) must also be given. If the “startFrame” argument is desired but input constraints were not used, the /dev/null argument may be given.

nFrames
This optional argument gives the desired number of frames that will be converted to PDB. The default behavior will convert all frames in the movie file. In order to specify this argument, the “constraints” and “startFrame” arguments must be given.

dFrames
This optional argument gives the number of frames between each converted snapshot. For instance, if a movie contains 2,000 snapshots, but only 200 snapshots spread evenly throughout the simulation are desired, a dFrames of 10 may be specified so that only frames 10, 20, 30, …, 2000 will be printed to the PDB output file. In order to specify this argument, the “constraints,” “startFrame,” and “nFrames” arguments must be given.

Once converted to PDB format, πDMD movies may be viewed using PyMol, VMD, or similar molecular visualization software. PDB files may also be used for standard structural and dynamic analysis.

To assure maximal efficiency and sampling in replica exchange simulations, the exchange rate between adjacent replicas should typically be between 0.25 and 0.5. At lower rates, the sampling of the system is negatively affected because the replicas are not exchanging at sufficient rate. At higher rates, the system is not able to equilibrate at the new temperature before being swapped again. To calculate the exchange rate for a given simulation:

Usage:

rx-rate.pl RXtemp.out

Here, RXtemp.out is the replica exchange output file, described in section 4.4.

8 πDMD Installation

8.1 Compatible Platforms

πDMD has been optimized on Linux platforms, and will be compatible with Mac and all x86 CPU UNIX-based platforms. Executables for these platforms are available on request to academic users or to commercial users with registered software.

8.2 Compilation

In order for the algorithm to function in a parallelized manner, the compilation of πDMD requires support of atomic built-ins, which are needed for thread synchronization. All recent versions of GCC and ICC compilers include support for atomic built-ins. If these built-ins are unavailable, Make will compile the single-threaded version of πDMD, and the program will use the non-parallelized algorithm.

πDMD utilizes AutoTools with standard makefiles in order to facilitate optimal compilation with little user input. For compilation, execute the following commands from the πDMD directory:

./configure

make

Further optimization is possible for advanced users.

8.3 Additional Documentation

Additional documentation is available for πDMD:

πDMD Tutorial – provides examples and sample files for setting up, performing, and analyzing simulations of several types of biological molecules.

πDMD QuickGuide – useful for diving straight into πDMD simulation, this guide provides a simple list of instructions for setting up and conducting basic πDMD simulations.

References

1. Proctor, E. A., Ding, F., and Dokholyan, N. V. "Discrete Molecular Dynamics." *Wiley Interdisciplinary Reviews: Computational Molecular Science*, 1:80-92, (2011).
2. Dokholyan, N. V., Buldyrev, S. V., Stanley, H. E., and Shakhnovich, E. I. "Molecular dynamics studies of folding of a protein-like model." *Folding and Design*, 3:577-587 (1998).
3. Ding, F., Tsao, D., Nie, H., and Dokholyan, N. V. "*Ab initio* folding of proteins with all-atom discrete molecular dynamics." *Structure*, 16:1010-1018, (2008).
4. Khan, M. A., and Herbordt, M. C. "Parallel discrete molecular dynamics simulation with speculation and in-order commitment." *Journal of Computational Physics,* 230:6563-6582 (2011).
5. Ding, F., Dokholyan, N. V., Buldyrev, S. V., Stanley, H. E., and Shakhnovich, E. I. "Direct molecular dynamics observation of protein folding transition state ensemble." *Biophysical Journal,* 83:3525-3532 (2002).
6. Chen, Y., Ding, F., and Dokholyan, N. V. "Fidelity of the protein structure reconstruction from inter-residue proximity constraints." *Journal of Physical Chemistry B,* 111:7432-7438 (2007).
7. Ding, F., Borreguero, J.M., Buldyrey, S.V., Stanley, H.E., and Dokholyan, N.V. "Mechanism for the α-helix to β-hairpin transition." *Proteins,* 53:220–228 (2003).
8. Ding, F., Buldyrev, S. V., Dokholyan, N. V. "Folding Trp-cage to NMR resolution native structure using a coarse-grained protein model." *Biophysical Journal,* 88:147-155 (2005).
9. Ding, F., Sharma, S., Chalasani, P., Demidov, V. V., Broude, N. E., and Dokholyan, N. V. "*Ab initio* RNA folding by discrete molecular dynamics: from structure prediction to folding mechanisms." *RNA* 14:1164–1173 (2008).

10. Sharma, S., Ding, F., and Dokholyan, N. V. “Multiscale modeling of nucleosome dynamics.” *Biophysical Journal* 92:1457–1470 (2002).
11. Gherghe, C. M., Leonard, C.W., Ding, F., Dokholyan, N. V., and Weeks, K.M. “Native-like RNA tertiary structures using a sequence-encoded cleavage agent and refinement by discrete molecular dynamics.” *Journal of the American Chemical Society* 131:2541–2546 (2009).
12. Davis, C. H., Nie, H., and Dokholyan, N. V. “Insights into thermophilic archaebacterial membrane stability from simplified models of lipid membranes.” *Physical Review E Statistical and Nonlinear Soft Matter Physics,* 75(5 Pt 1):051921 (2007).
13. Alder, B. J., and Wainwright, T. E. “Studies in molecular dynamics. I. General method.” *Journal of Chemical Physics*, 31:459-466 (1959).
14. Ding, F., and Dokholyan, N. V. "Emergence of protein fold families through rational design." *Public Library of Science Computational Biology*, 2:e85, (2006).
15. Yin, S., Biedermannova, L., Vondrasek, J., and Dokholyan, N. V. "MedusaScore: An accurate force-field based scoring function for virtual drug screening." *Journal of Chemical Information and Modeling*, 48:1656-1662, (2008).
16. Ding, F., Yin, S., and Dokholyan, N. V. "Rapid flexible docking using a stochastic rotamer library of ligands." *Journal of Chemical Information and Modeling*, 50:1623-1632 (2010).
17. Brooks, B. R., Bruccoleri, R. E., Olafson, B. D., States, D. J., Swaminathan, S., and Karplus, M. “CHARMM - A Program for Macromolecular Energy, Minimization, and Dynamics Calculations.” *Journal of Computational Chemistry,* 4:187-217 (1983).
18. Lazaridis, T., and Karplus, M. “Effective Energy Function for Proteins in Solution.” *Proteins: Structure, Function, and Genetics* 35:133-152 (1999).
19. Kortemme, T., Baker, D. “A simple physical model for binding energy hot spots in protein-protein complexes.” *Proceedings of the National Academy of Sciences U.S.A,* 99:14116-14121 (2002).
20. Okamoto, Y. “Generalized-ensemble algorithms: enhanced sampling techniques for Monte Carlo and molecular dynamics simulations.” *Journal of Molecular Graphic Modeling,* 22:425–439 (2004).
21. Zhou, R., Berne, B. J., and Germain, R. “The free energy landscape for beta hairpin folding in explicit water.” *Proceedings of the National Academy of Sciences U.S.A,* 98:14931– 14936 (2001).

22. Andersen, H. C. “Molecular-dynamics simulations at constant pressure and/or temperature.” *Journal of Chemical Physics,* 72:2384–2393 (1980).
23. Taketomi, H., Ueda, Y., Gō, N. “Studies on protein folding, unfolding and fluctuations by computer simulation. I. The effect of specific amino acid sequence represented by specific inter-unit interactions.” *International Journal of Peptide and Protein Research*, 7:445–459 (1975).
24. Gō, N. “Theoretical studies of protein folding.” *Annual Review of Biophysics and Bioengineering*, 12:183–210 (1983).