foldamers Documentation

Release 0.0.1

Garrett A. Meek, Lenny T. Fobe, Michael R. Shirts

Apr 14, 2019

CONTENTS

This documentation is generated automatically using Sphinx, which reads all docstring-formatted comments from Python functions in the 'foldamers' repository. (See foldamers/doc for Sphinx source files.)

CHAPTER

ONE

COARSE GRAINED MODEL UTILITIES

This page details the functions and classes in src/cg_model/cgmodel.py

1.1 'cgmodel' class for OpenMM simulation

Shown below is a detailed description of the 'cgmodel' class object, which contains all information about a coarse grained model.

1.2 Other coarse grained model utilities

CHAPTER

TWO

COARSE GRAINED MODEL PARAMETER ANALYSIS UTILITIES

This page details the modules, functions, and classes in src/parameter_analysis

2.1 Parameter sampling protocols

CHAPTER

THREE

THERMODYNAMIC ANALYSIS TOOLS FOR COARSE GRAINED MODELING

This page details the functions and classes in src/thermo

3.1 Heat capacity example with pymbar

Shown below are functions/tools used in order to calculate the heat capacity with pymbar.

CHAPTER

FOUR

UTILITIES FOR THE 'FOLDAMERS' PACKAGE

This page details the functions and classes in src/util.

4.1 Input/Output options (src/utilities/iotools.py)

Shown below is a detailed description of the input/output options for the foldamers package.

4.2 Utilities and random functions (src/utilities/util.py)

util.**append_position**(*positions*, *new_coordinate*)

Updates a set of input coordinates with 'new_coordinate' in the cartesian coordinate direction indexted by 'direction'.

new_coordinate: Cartesian coordinates for a particle (np.array(float * unit (length = 3)))

direction: Cartesian direction index for particle placement, where: x=0,y=1,z=2. (integer)

trial_coordinates: Existing cartesian coordinates for the particle we are updating. (np.array(float * unit (length = 3))) Optional, default = None

trial_coordinates: Updated coordinates for the particle.

util.**assign_backbone_beads**(*positions*, *monomer_start*, *backbone_length*, *sidechain_length*, *sidechain_positions*, *bond_length*)

Assign random position for a backbone bead

positions: Positions for all beads in the coarse-grained model. (np.array(num_beads x 3))

monomer_start: Index of the bead to which we will bond this new backbone bead. (integer)

backbone_length: Number of beads in the backbone portion of each (individual) monomer (integer), default = 1

sidechain_length: Number of beads in the sidechain portion of each (individual) monomer (integer), default = 1

sidechain_positions: List of integers defining the backbone bead indices upon which we will place the sidechains, default = [0] (Place a sidechain on the backbone bead with index "0" (first backbone bead) in each (individual) monomer

bond_length: Bond length for all beads that are bonded, (float * simtk.unit.distance) default = 1.0 * unit.angstrom

positions: Positions for all beads in the coarse-grained model. (np.array(num_beads x 3))

util.**assign_position**(*positions*, *bond_length*, *parent_index=-1*)
Assign random position for a bead

positions: Positions for all beads in the coarse-grained model. (np.array(num_beads x 3))

bond_length: Bond length for all beads that are bonded, (float * simtk.unit.distance) default = 1.0 * unit.angstrom

positions: Positions for all beads in the coarse-grained model. (np.array(num_beads x 3))

util.**assign_sidechain_beads**(*positions*, *sidechain_length*, *bond_length*)
Assign random position for all sidechain beads

positions: Positions for all beads in the coarse-grained model. (np.array(num_beads x 3))

sidechain_length: Number of beads in the sidechain portion of each (individual) monomer (integer), default = 1

bond_length: Bond length for all beads that are bonded, (float * simtk.unit.distance) default = 1.0 * unit.angstrom

positions: Positions for all beads in the coarse-grained model. (np.array(num_beads x 3))

util.**attempt_move**(*parent_coordinates*, *bond_length*)
Given a set of cartesian coordinates, assign a new particle a distance of 'bond_length' away in a random direction.

parent_coordinates: Positions for a single particle, away from which we will place a new particle a distance of 'bond_length' away. (np.array(float * unit.angstrom (length = 3)))

bond_length: Bond length for all beads that are bonded, (float * simtk.unit.distance) default = 1.0 * unit.angstrom

trial_coordinates: Positions for a new trial particle (np.array(float * unit.angstrom (length = 3)))

util.**collisions**(*distances*, *bond_length*)
Determine whether there are any collisions between non-bonded particles, where a "collision" is defined as a distance shorter than the user-provided 'bond_length'.

distances: List of the distances between all nonbonded particles. (list (float * simtk.unit.distance (length = # nonbonded_interactions)))

bond_length: Bond length for all beads that are bonded, (float * simtk.unit.distance) default = 1.0 * unit.angstrom

collision: Logical variable stating whether or not the model has bead collisions. default = False

util.**distance**(*positions_1*, *positions_2*)
Construct a matrix of the distances between all particles.

positions_1: Positions for a particle (np.array(length = 3))

positions_2: Positions for a particle (np.array(length = 3))

distance (float * unit)

util.**distance_matrix**(*positions*)
Construct a matrix of the distances between all particles.

positions: Positions for an array of particles. (np.array(num_particles x 3))

distance_matrix: Matrix containing the distances between all beads. (np.array(num_particles x 3))

util.**first_bead**(*positions*)
Determine if we have any particles in 'positions'

positions: Positions for all beads in the coarse-grained model. (np.array(float * unit (shape = num_beads x 3)))

first_bead: Logical variable stating if this is the first particle.

util.**get_move**(*direction*, *step*)
Given the cartesian coordinates for a particle ('move'), a 'step' (distance), and a 'direction' (Index denoting x,y,z Cartesian direction), update the coordinates for the particle.

direction: Cartesian directions in which we have attempted a particle placement, where: x=0,y=1,z=2. (integer)

step: Number to add/subtract to the cartesian coordinates for direction 'direction' in 'move' (float * simtk.unit.distance)

move: Updated positions for the particle (np.array(float * unit.angstrom (length = 3)))

util.**non_bonded_distances**(*new_coordinates*, *existing_coordinates*)
Calculate the distances between a trial particle ('new_coordinates') and all existing particles ('existing_coordinates').

new_coordinates: Positions for a single trial particle (np.array(float * unit.angstrom (length = 3)))

existing_coordinates: Positions for a single trial particle (np.array(float * unit.angstrom (shape = num_particles x 3)))

distances: List of the distances between all nonbonded particles. (list (float * simtk.unit.distance (length = # nonbonded_interactions)))

util.**random_positions**(*polymer_length*, *backbone_length*, *sidechain_length*, *sidechain_positions*, *bond_length*, *sigma*)
Assign random positions for all beads in a coarse-grained polymer.

polymer_length: Number of monomer units (integer), default = 8

backbone_length: Number of beads in the backbone portion of each (individual) monomer (integer), default = 1

sidechain_length: Number of beads in the sidechain portion of each (individual) monomer (integer), default = 1

sidechain_positions: List of integers defining the backbone bead indices upon which we will place the sidechains, default = [0] (Place a sidechain on the backbone bead with index "0" (first backbone bead) in each (individual) monomer

bond_length: Bond length for all beads that are bonded, (float * simtk.unit.distance) default = 1.0 * unit.angstrom

sigma: Non-bonded bead Lennard-Jones interaction distances, (float * simtk.unit.distance) default = 8.4 * unit.angstrom

bb_bond_length: Bond length for all bonded backbone beads, (float * simtk.unit.distance) default = 1.0 * unit.angstrom

bs_bond_length: Bond length for all backbone-sidechain bonds, (float * simtk.unit.distance) default = 1.0 * unit.angstrom

ss_bond_length: Bond length for all beads within a sidechain, (float * simtk.unit.distance) default = 1.0 * unit.angstrom

positions: Positions for all beads in the coarse-grained model. (np.array(num_beads x 3))

util.**random_sign**(*number*)
Returns 'number' with a random sign.

number: float

number

util.**unit_sqrt**(*simtk_quantity*)
Returns the square root of a simtk 'Quantity'.

simtk_quantity: A 'Quantity' object, as defined in simtk. (float * unit)

sqrt: Square root of a simtk_quantity.

util.**update_trial_coordinates**(*move*, *trial_coordinates=None*)
Updates 'trial_coordinates by adding the coordinates in 'move'.

move: Cartesian coordinates for a particle (np.array(float * unit (length = 3)))

trial_coordinates: Existing cartesian coordinates for the particle we are updating. (np.array(float * unit (length = 3))) Optional, default = None

new_coordinates: Updated coordinates for the particle.

CHAPTER

FIVE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

INDEX