foldamers Documentation

Release 0.0

Shirts research group

Apr 29, 2019

CONTENTS

class cg_model.cgmodel.**CGModel**(*positions=None, polymer_length=12, backbone_length=1, sidechain_length=1, sidechain_positions=[0], mass=Quantity(value=12.0, unit=dalton), sigma=Quantity(value=8.4, unit=angstrom), epsilon=Quantity(value=0.5, unit=kilocalorie/mole), bond_length=Quantity(value=1.0, unit=angstrom), bond_force_constant=990000.0, bb_bond_length=Quantity(value=1.0, unit=angstrom), bs_bond_length=Quantity(value=1.0, unit=angstrom), ss_bond_length=Quantity(value=1.0, unit=angstrom), charge=Quantity(value=0.0, unit=elementary charge), constrain_bonds=False*)

Construct a coarse grained model.

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

mass: Mass of coarse grained beads (float * simtk.unit.mass) default = 12.0 * unit.amu

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

epsilon: Non-bonded bead Lennard-Jones interaction strength, (float * simtk.unit.energy) default = 0.5 * unit.kilocalorie_per_mole

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

bond_force_constant: Bond force constant for all beads that are bonded, (float) default = 9.9e5 kJ/mol/A^2

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

charge: Charge for all beads (float * simtk.unit.charge) default = 0.0 * unit.elementary_charge

polymer_length backbone_length sidechain_length sidechain_positions mass sigma epsilon bond_length bond_force_constant bb_bond_length bs_bond_length ss_bond_length charge num_beads positions system

constrain_bonds = None

Initialize new (coarse grained) particle types:

positions = None

Initialize attributes of our coarse grained model.

utilities.util.**append_position**(*positions*, *new_coordinates*)

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

new_coordinates: 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.

utilities.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))

utilities.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))

utilities.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))

utilities.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)))

utilities.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

utilities.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)

utilities.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))

utilities.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.

utilities.util.**get_move**(*trial_coordinates*, *move_direction*, *distance*, *bond_length*, *finish_bond=False*)
Given a 'move_direction', a current distance, and a target 'bond_length' (Index denoting x,y,z Cartesian direction), update the coordinates for the particle.

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

move_direction: Cartesian direction in which we will attempt a particle placement, where: x=0, y=1, z=2. (integer)

distance: Current distance from parent particle (float * simtk.unit.distance)

bond_length: Target bond_length for particle placement. (float * simtk.unit.distance)

finish_bond: Logical variable determining how we will update the coordinates for this particle.

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

utilities.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)))

utilities.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))

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

number: float

number

utilities.util.**single_bead**(*positions*)
Determine if we have one particle in positions

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

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

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

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

answer: Square root of a simtk_quantity.

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

move: Cartesian coordinates for a new particle placement (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.

utilities.iotools.**write_pdbfile**(*CGModel*, *filename*)
Writes the positions in 'CGModel' to the file 'filename'.

CGModel: Coarse grained model class object

filename: Path to the file where we will write PDB coordinates.

CHAPTER

ONE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

INDEX