Coarse-grained OpenMM Documentation

Release 0.0.1

Garrett A. Meek
Lenny T. Fobe

Research group of Michael R. Shirts

Dept. of Chemical and Biological Engineering
University of Colorado Boulder

Jul 30, 2019

CONTENTS

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

CHAPTER

ONE

OPENMM SIMULATION() PROTOCOLS FOR COARSE GRAINED MODELING

This page details the functions in cg_openmm/src/build/cg_build.py.

build.cg_build.**add_force**(*cgmodel*, *force_type=None*)

build.cg_build.**add_new_elements**(*cgmodel*)

Adds new coarse grained particle types to OpenMM

cgmodel: CGModel() class object

list_of_masses: List of masses for the particles we want to add to OpenMM

build.cg_build.**build_mm_force**(*sigma*, *epsilon*, *charge*, *num_beads*, *cutoff=Quantity(value=1*, *unit=nanometer)*)

Build an OpenMM 'Force' for the non-bonded interactions in our model.

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

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

charge: Charge for all beads (float * simtk.unit.charge)

cutoff: Cutoff distance for nonbonded interactions (float * simtk.unit.distance)

num_beads: Total number of beads in our coarse grained model (integer)

build.cg_build.**build_system**(*cgmodel*)

Builds an OpenMM System() class object, given a CGModel() class object as input.

cgmodel: CGModel() class object

system: OpenMM System() class object

build.cg_build.**build_topology**(*cgmodel*, *use_pdbfile=False*)

Construct an OpenMM topology for our coarse grained model

polymer_length: Number of monomers in our coarse grained model (integer)

backbone_length: Number of backbone beads on individual monomers in our coarse grained model, (integer)

sidechain_length: Number of sidechain beads on individual monomers in our coarse grained model, (integer)

build.cg_build.**get_num_forces**(*cgmodel*)
Given a coarse grained model class object, this function dtermines how many forces we are including when evaluating its energy.

build.cg_build.**test_force**(*cgmodel*, *force*, *force_type=None*)

build.cg_build.**test_forces**(*cgmodel*)

build.cg_build.**verify_system**(*cgmodel*)
Given a coarse grained model class object, this function confirms that its OpenMM system object is configured correctly.

build.cg_build.**verify_topology**(*cgmodel*)
Verify the OpenMM topology for our coarse grained model

build.cg_build.**write_xml_file**(*cgmodel*, *xml_file_name*)

CHAPTER

TWO

OPENMM SIMULATION TOOLS FOR COARSE GRAINED MODELING

This page details the functions in cg_openmm/src/simulation/.

2.1 Replica exchange simulation tools for coarse grained modeling

simulation.rep_exch.**get_replica_energies**(*simulation_steps*, *num_replicas*, *replica_exchange_storage_file*, *exchange_attempts*)

Parameters

- **simulation_steps** (*integer*) – The number of steps to take during each replica exchange simulation
- **num_replicas** (*integer*) – The number of temperature replicas for which we will perform molecular dynamics simulations
- **replica_exchange_storage_file** (*string*) – The path to a NETCDF file containing the compressed output from all replica exchange simulation runs
- **exchange_attempts** (*integer*) – The number of times that temperature replica exchanges were attempted

replica_energies: List(List(float * simtk.unit.energy for simulation_steps) for num_replicas)
List of dimension num_replicas X simulation_steps, which gives the energies for all replicas at all simulation steps

simulation.rep_exch.**replica_exchange**(*topology, system, positions, temperature_list=[Quantity(value=250.0, unit=kelvin), Quantity(value=260.0, unit=kelvin), Quantity(value=270.0, unit=kelvin), Quantity(value=280.0, unit=kelvin), Quantity(value=290.0, unit=kelvin), Quantity(value=300.0, unit=kelvin), Quantity(value=310.0, unit=kelvin), Quantity(value=320.0, unit=kelvin), Quantity(value=330.0, unit=kelvin), Quantity(value=340.0, unit=kelvin)], simulation_time_step=None, total_simulation_time=Quantity(value=1.0, unit=picosecond), output_data='output.nc', print_frequency=100, verbose=False, verbose_simulation=False, exchange_attempts=None, test_time_step=False*)

Construct an OpenMM simulation object for our coarse grained model.

Parameters

- **topology** (*OpenMM Topology() class object*) – An OpenMM object which contains information about the bonds and constraints in a molecular model
- **system** (*OpenMM System() class object*) – An OpenMM object which contains information about the forces and particle properties in a molecular model
- **positions** (*np.array('num_beads' x 3 , (float * simtk.unit.distance))*) – Contains the positions for all particles in a model
- **temperature_list** – List of temperatures for which to perform replica exchange simulations, default = [(300.0 * unit.kelvin).__add__(i * unit.kelvin) for i in range(-20,100,10)]
- **simulation_time_step** (*float * simtk.unit*) – Simulation integration time step, default = None

- **total_simulation_time** (*float * simtk.unit*) – Total simulation time
- **output_data** (*string*) – Name of NETCDF file where we will write data from replica exchange simulations

replica_energies: List(List(float * simtk.unit.energy for simulation_steps) for num_replicas)
List of dimension num_replicas X simulation_steps, which gives the energies for all replicas at all simulation steps

2.2 General simulation tools for coarse grained modeling

simulation.tools.**build_mm_simulation**(*topology*, *system*, *positions*, *temperature=Quantity(value=300.0, unit=kelvin)*, *simulation_time_step=None*, *total_simulation_time=Quantity(value=1.0, unit=picosecond)*, *output_pdb='output.pdb'*, *output_data='output.dat'*, *print_frequency=100*, *test_time_step=False*)

Construct an OpenMM simulation object for our coarse grained model.

topology: OpenMM topology object

system: OpenMM system object

positions: Array containing the positions of all beads in the coarse grained model (np.array('num_beads' x 3 , (float * simtk.unit.distance))

temperature: Simulation temperature (float * simtk.unit.temperature)

simulation_time_step: Simulation integration time step (float * simtk.unit.time)

total_simulation_time: Total simulation time (float * simtk.unit.time)

output_data: Name of output file where we will write the data from this simulation (string)

print_frequency: Number of simulation steps to skip when writing data to 'output_data' (integer)

simulation.tools.**get_mm_energy**(*topology*, *system*, *positions*)

Get the OpenMM potential energy for a system, given a topology and set of positions.

topology: OpenMM topology object

system: OpenMM system object

positions: Array containing the positions of all beads in the coarse grained model (np.array('num_beads' x 3 , (float * simtk.unit.distance))

simulation.tools.**get_simulation_time_step**(*topology*, *system*, *positions*, *temperature*, *total_simulation_time*, *time_step_list=None*)

Determine a valid simulation time step for our coarse grained model.

simulation: OpenMM simulation object

time_step_list: List of time steps for which to attempt a simulation in OpenMM.

time_step: A time step that was successful for our simulation object.

simulation.tools.**minimize_structure**(*topology*, *system*, *positions*, *temperature=Quantity(value=0.0, unit=kelvin)*, *simulation_time_step=None*, *total_simulation_time=Quantity(value=1.0, unit=picosecond)*, *output_pdb='minimum.pdb'*, *output_data='minimization.dat'*, *print_frequency=1*)

Construct an OpenMM simulation object for our coarse grained model.

topology: OpenMM topology object

system: OpenMM system object

positions: Array containing the positions of all beads in the coarse grained model (np.array('num_beads' x 3 , (float * simtk.unit.distance))

temperature: Simulation temperature (float * simtk.unit.temperature)

simulation_time_step: Simulation integration time step (float * simtk.unit.time)

output_data: Name of output file where we will write the data from this simulation (string)

print_frequency: Number of simulation steps to skip when writing data to 'output_data' (integer)

CHAPTER

THREE

UTILITIES FOR COARSE GRAINED MODELING IN OPENMM

This page details the functionality of utilities in cg_openmm/src/utilities/util.py.

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.**get_box_vectors**(*box_size*)
Assign all side lengths for simulation box.

box_size: Simulation box length (float * simtk.unit.length)

utilities.util.**set_box_vectors**(*system*, *box_size*)
Build a simulation box.

system: OpenMM system object

box_size: Simulation box length (float * simtk.unit.length)

CHAPTER

FOUR

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

b

s

u

INDEX