

---

# **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**

**Jun 03, 2019**

# CONTENTS

<b>1</b>	<b>OpenMM Simulation() protocols for coarse grained modeling</b>	<b>2</b>
<b>2</b>	<b>OpenMM simulation tools for coarse grained modeling</b>	<b>4</b>
2.1	Replica exchange simulation tools for coarse grained modeling . . . . .	5
2.2	General simulation tools for coarse grained modeling . . . . .	6
<b>3</b>	<b>Utilities for coarse grained modeling in OpenMM</b>	<b>7</b>
<b>4</b>	<b>Indices and tables</b>	<b>8</b>
	<b>Python Module Index</b>	<b>9</b>
	<b>Index</b>	<b>10</b>

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.)

## OPENMM SIMULATION() PROTOCOLS FOR COARSE GRAINED MODELING

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

`build.cg_build.add_new_elements(cgmodel, list_of_masses)`

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_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)`

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` )

`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*)

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` )

## 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.replica_exchange(topology, system, positions, temper-
                                     ature_list=[Quantity(value=280.0,
                                     unit=kelvin),           Quan-
                                     tity(value=290.0,           unit=kelvin),
                                     Quantity(value=300.0,
                                     unit=kelvin),           Quan-
                                     tity(value=310.0,           unit=kelvin),
                                     Quantity(value=320.0,
                                     unit=kelvin),           Quan-
                                     tity(value=330.0,           unit=kelvin),
                                     Quantity(value=340.0,
                                     unit=kelvin),           Quan-
                                     tity(value=350.0,           unit=kelvin),
                                     Quantity(value=360.0,
                                     unit=kelvin),           Quan-
                                     tity(value=370.0,           unit=kelvin),
                                     Quantity(value=380.0,
                                     unit=kelvin),           Quan-
                                     tity(value=390.0,           unit=kelvin)],
                                     simulation_time_step=None,   to-
                                     tal_simulation_time=Quantity(value=1.0,
                                     unit=picosecond),           out-
                                     put_pdb='output.pdb',
                                     output_data='output.nc',
                                     print_frequency=100,
                                     verbose=False,           ver-
                                     bose_simulation=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 per ( 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 )

## 2.2 General simulation tools for coarse grained modeling

```
simulation.tools.get_simulation_time_step(topology, system, positions,  
                                         temperature, time_step_list,  
                                         total_simulation_time)
```

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, tem-  
                                  perature=Quantity(value=0.0,  
                                  unit=kelvin), simula-  
                                  tion_time_step=None, to-  
                                  total_simulation_time=Quantity(value=1.0,  
                                  unit=picosecond), out-  
                                  put_pdb='minimum.pdb', out-  
                                  put_data='minimization.dat',  
                                  print_frequency=10)
```

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 )



## 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` )

## INDICES AND TABLES

- genindex
- modindex
- search

## PYTHON MODULE INDEX

### **b**

`build.cg_build`, 2

### **s**

`simulation.rep_exch`, 5

`simulation.tools`, 6

### **u**

`utilities.util`, 7

## INDEX

### A

`add_new_elements()` (in module *build.cg\_build*), 2

### B

`build.cg_build(module)`, 2

`build_mm_force()` (in module *build.cg\_build*), 2

`build_mm_simulation()` (in module *build.cg\_build*), 2

`build_system()` (in module *build.cg\_build*), 3

`build_topology()` (in module *build.cg\_build*), 3

### D

`distance()` (in module *utilities.util*), 7

### G

`get_box_vectors()` (in module *utilities.util*), 7

`get_simulation_time_step()` (in module *simulation.tools*), 6

### M

`minimize_structure()` (in module *simulation.tools*), 6

### R

`replica_exchange()` (in module *simulation.rep\_exch*), 5

### S

`set_box_vectors()` (in module *utilities.util*), 7

`simulation.rep_exch(module)`, 5

`simulation.tools(module)`, 6

### U

`utilities.util(module)`, 7