Terphenyl folding Documentation

Release 0.0

Garrett A. Meek
Theodore L. Fobe
Benjamin J. Coscia

Research group of Professor Michael R. Shirts

Dept. of Chemical and Biological Engineering
University of Colorado Boulder

Sep 25, 2019

CONTENTS

1 Terphenyl folding simulation utilities 2

2 Terphenyl folding analysis tools 6

Python Module Index 8

Index 9

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

CHAPTER

ONE

TERPHENYL FOLDING SIMULATION UTILITIES

Shown below are tools that allow simulation of terphenyl oligomers.

simulation.**adjust_solvent_density**(*solvent_file*, *target_solvent_density*)

simulation.**build_directories**(*polymer_name*, *polymer_length*, *fresh_run=False*)

Given a set of input strings, this function builds the directories that are needed to perform GROMACS simulations with terphenyl oligomers.

Parameters

- **polymer_name** (*str*) – The name of the polymer
- **polymer_length** (*str*) – The length of the polymer we are modeling (in monomer units)
- **run_directory** (*str*) – The directory where simulations will be run
- **fresh_run** (*Logical*) – A logical variable determining whether old run files should be removed.

Returns

- run_directory (str) - The path to a directory where simulations will be run.
- pdb_file (str) - The path to pdb file that will be used for simulations.
- solvent_file (str) - The path to a file containing a box of solvent molecules
- topology_file (str) - The path to a file containing the topology for the pdb_file.

simulation.**calculate_solvent_density**(*solvent_file*)

Given an input file containing solvent molecules, this function computes the solvent density, in

simulation.**compress_large_files**(*directory*, *size_threshold=100000000.0*)

simulation.**equilibrate**(*run_directory*, *input_gro_file*)
Given an input system, this function performs pressure (Berendsen) equilibration.

Parameters

- **run_directory** (*str*) – The path to a directory where the equilibration will be run.
- **input_gro_file** (*str*) – The path to a .gro file that will be used for the equilibration run.

Returns

- output_trajectory (str) - The path to a PDB file containing the output for the equilibration run.

simulation.**get_box_volume**(*solvent_file*)
Given an input file containing data for a simulation box of solvent molecules, this function calculates the box volume.

simulation.**get_num_solvent_molecules**(*solvent_file*)

simulation.**get_terphenyl_top_directory**()

simulation.**make_topology**(*polymer_code*, *num_solvent_molecules=0*)

simulation.**minimize**(*run_directory*, *topology*, *input_structure*, *polymer_code*)
Given a GROMACS .mdp MD commands file, a topology, and an input structure, this function performs a mimimization.

Parameters

- **run_directory** (*str*) – The path to a directory where the minimization will be run.
- **topology** (*str*) – The path to a topology file
- **input_structure** (*type*) – The path to an input structure to minimize.

Returns

- minimized_pdb_file (str) - The path to a PDB file for the minimized structure.

simulation.**parameterize**(*param_directory*, *pdb_file*, *topology_file*, *polymer_code*, *polymer_length*)
Given a directory path, PDB file, and a topology file, this function parameterizes the structure with GAFF.

Parameters

- **param_directory** (*str*) – The path to a directory where intermediate and output parameterization files will be written.

- **pdb_file** (*str*) – The path to a PDB file containing data for the structure that will be parameterized.
- **topology_file** (*str*) – The path to a file containing the topology for the structure that will be parameterized.
- **polymer_code** (*str (length=3)*) – A three-letter code for the polymer (required for GAFF parameterization, in place of residue codes)
- **polymer_length** (*str*) – The length of the input structure (number of monomers).

Returns

- solute_gro_file (str) - Path to the GAFF-parameterized .gro file for the input structure.

simulation.**remove_random_molecules**(*solvent_file*, *num_molecules_to_remove*)

simulation.**replace**(*file*, *original_text*, *replacement_text*)

Given a file, a target search string, and a replacement string, this function replaces the text in 'file'.

Parameters

- **file** (*file*) – A file containing the text that will be replaced.
- **original_text** (*str*) – Text that will be replaced.
- **replacement_text** (*str*) – Text that will be used to replace the original text.

simulation.**simulate**()

simulation.**solvate**(*solvation_directory*, *solute_gro_file*, *solute_topology_file*, *solvent_file*, *polymer_code*, *solvent_density=None*)

Given solute and solvent .gro files, this function produces a combined, solvated .gro file.

Parameters

- **solvation_directory** (*str*) – The path to a directory where solvation will be performed.
- **solute_gro_file** (*str*) – The path to a .gro file for the solute
- **solute_topology_file** (*str*) – The path to a topology file for the solute.
- **solvent_file** (*str*) – The path to a .gro file containing solvent
- **solvent_density** (*float*) – The density for the solvent box.

Returns

- solvated_gro_file (str) - The path to a .gro file containing the combined, solvated system.

CHAPTER

TWO

TERPHENYL FOLDING ANALYSIS TOOLS

Shown below are functions/tools that allow analysis of terphenyl oligomer simulation results.

analysis.**construct_selector**(*atom_list*, *res_dict*, *n_residues*)
Since our molecule is all in one residue, we need some interesting ways to extract residues, instead of a simple select string

atom_list [list] String name of base atoms in the first residue

res_dict [dict] Dictionary telling how many atoms of a given element are in each residue

analysis.**get_end_to_end_distance**(*structure*, *atom_pair*)

analysis.**get_internal_coordinate_definitions**(*structure*, *polymer_name*, *polymer_length*)
structure: pdb file path

polymer_name: string (ie: “o-terphenyl”)

polymer_length: string (ie: “monomer”)

analysis.**get_phenyl_carbon_indices**(*polymer_name*, *polymer_length*)
polymer_name: string (ie: “o-terphenyl”)

polymer_length: string (ie: “monomer”)

phenyl_indices_list: List(dict(‘1’: [phenyl-1 carbon indices], ‘2’: [phenyl-2 carbon indices], ‘3’: [phenyl-3 carbon indices]))

analysis.**get_phenyl_centers_of_mass**(*structure*, *polymer_name*, *polymer_length*)
structure: pdb file path

polymer_name: string (ie: “o-terphenyl”)

polymer_length: string (ie: “monomer”)

analysis.**get_terminal_atoms**(*structure_file*)

analysis.**read_trajectory**(*structure*, *trajectory*)

structure: pdb file path

trajectory: xtc file path

PYTHON MODULE INDEX

a

analysis, 6

s

simulation, 2

INDEX

A

adjust_solvent_density() (*in module simulation*), 2
analysis (*module*), 6

B

build_directories() (*in module simulation*), 2

C

calculate_solvent_density() (*in module simulation*), 2
compress_large_files() (*in module simulation*), 2
construct_selector() (*in module analysis*), 6

E

equilibrate() (*in module simulation*), 2

G

get_box_volume() (*in module simulation*), 3
get_end_to_end_distance() (*in module analysis*), 6
get_internal_coordinate_definitions() (*in module analysis*), 6
get_num_solvent_molecules() (*in module simulation*), 3
get_phenyl_carbon_indices() (*in module analysis*), 6
get_phenyl_centers_of_mass() (*in module analysis*), 6
get_terminal_atoms() (*in module analysis*), 6
get_terphenyl_top_directory() (*in module simulation*), 3

M

make_topology() (*in module simulation*), 3
minimize() (*in module simulation*), 3

P

parameterize() (*in module simulation*), 3

R

read_trajectory() (*in module analysis*), 6
remove_random_molecules() (*in module simulation*), 4
replace() (*in module simulation*), 4

S

simulate() (*in module simulation*), 4
simulation (*module*), 2
solvate() (*in module simulation*), 4