Trajectome

Release 1.0

Neelesh Soni

Feb 10, 2025

CONTENTS:

CHAPTER

ONE

TRAJECTOME

1.1 Trajectome

1.1.1 Submodules

1.1.2 src.Interaction_Class module

Author: Neelesh Soni, neelesh@salilab.org, neeleshsoni03@gmail.com Date: April 5, 2024

src.Interaction_Class.**logger**

Description

Type
TYPE

class src.Interaction_Class.**Interaction**(*system*)

Bases: object

A class to represent interactions between proteins within a simulation system.

The *Interaction* class is responsible for defining and adding various types of restraints between proteins to model their interactions in a Brownian dynamics simulation.

system

The simulation system in which the interactions are defined.

Type
System

add_binding_restraint(*prot1_tuple*, *prot2_tuple*, *name*, *mean_dist*, *kappa*)

Adds a harmonic binding restraint between two proteins.

This function defines a distance-based harmonic restraint between two protein particles, ensuring they remain within a defined mean distance with a given kappa value.

Parameters

- **prot1_tuple** (*tuple*) – A tuple containing the first protein name and its index.
- **prot2_tuple** (*tuple*) – A tuple containing the second protein name and its index.
- **name** (*str*) – The name of the restraint.
- **mean_dist** (*float*) – The expected mean distance between the proteins.
- **kappa** (*float*) – The force constant for the harmonic function.

Returns
None

`add_distance_restraint`(*prot1*, *prot2*, *dist*, *k*)

Adds a distance-based restraint between two proteins.

This function creates a distance restraint to enforce a specific separation between two proteins. The restraint applies a force that maintains the proteins at a given distance with the specified force constant.

Parameters

- **prot1** (Protein) – The first protein.
- **prot2** (Protein) – The second protein.
- **dist** (*float*) – The target distance for the restraint.
- **k** (*float*) – The force constant for the restraint.

Returns
None

1.1.3 src.Protein_Class module

Author: Neelesh Soni, neelesh@salilab.org, neeleshsoni03@gmail.com Date: April 5, 2024

This module defines the *Protein* class used for protein simulations.

`src.Protein_Class.logger`

Logger for the module.

Type
logging.Logger

`class src.Protein_Class.Protein`(*model*, *name*, *center=[1, 1, 1]*, *radius=30*, *mass=1000*, *diffcoff=None*, *color=0*)

Bases: `object`

A class to represent a protein in a simulation.

model

The IMP model.

Type
IMP.Model

name

Name of the protein.

Type
str

center

Coordinates of the protein's center.

Type
list

radius

Radius of the protein.

Type
int

`mass`

Mass of the protein.

Type
int

`diffcoff`

Diffusion coefficient of the protein.

Type
float

`color`

Color index for display.

Type
int

`protp`

The main particle representing the protein.

Type
IMP.Particle

`hier`

Hierarchy setup for the protein.

Type
IMP.atom.Hierarchy

`prb`

Rigid body of the protein.

Type
IMP.core.RigidBody

`mol`

Molecule representation of the protein.

Type
IMP.atom.Molecule

`dif`

Diffusion setup for the protein.

Type
IMP.atom.Diffusion

`create_rigid_body_protein()`

Creates a rigid body representation of the protein.

This method sets up the protein structure by combining residues into a rigid body and configuring necessary attributes such as diffusion coefficient.

Parameters
`None`

Returns
None

class src.Protein_Class.**ProteinStructure**(*model*, *state*, *h_root*, *name*, *pdbfile*, *pdb_multimodel=False*, *resolution=50*, *color=0*, *centerize=False*, *diffcoff=None*, *rot_diffcoff_scale=None*)

Bases: `object`

A class representing a protein structure in the simulation.

The *ProteinStructure* class loads protein structural information from PDB or CIF files, processes it into a hierarchical representation, and applies coarse-grained modeling for molecular simulations. It also supports setting up diffusion properties and rigid-body representations.

model

The IMP model used for simulation.

Type
IMP.Model

state

The state of the protein in the simulation.

Type
IMP.pmi.topology.State

h_root

Root hierarchy of the system.

Type
IMP.atom.Hierarchy

name

The name of the protein.

Type
str

pdbfile

Path to the PDB file containing the protein structure.

Type
str

multi_model

Whether the PDB file contains multiple models.

Type
bool

resolution

Resolution level for coarse-graining the protein.

Type
int

color

Color index for visualization.

Type
int

`centerize`

Whether to centerize the protein coordinates.

Type
bool

`diffcoff`

Translational diffusion coefficient of the protein.

Type
float

`rot_diffcoff_scale`

Scaling factor for rotational diffusion.

Type
float

`protein`

IMP particle representing the protein.

Type
IMP.Particle

`hier`

Hierarchical representation of the protein structure.

Type
IMP.atom.Hierarchy

`mol`

IMP molecule representation for managing residues.

Type
IMP.atom.Molecule

`prb`

Rigid body representation of the protein.

Type
IMP.core.RigidBody

`amino_acid_radii`

Dictionary mapping amino acid names to their default radii.

Type
dict

`amino_acid_mass`

Dictionary mapping amino acid names to their default masses.

Type
dict

`All_Residues`

Ordered dictionary storing protein residues.

Type
OrderedDict

`Coarse_Residues`

List of coarse-grained residue fragments.

Type
list

`Protein_Residues`

List of IMP particles representing residues.

Type
list

`Tran_dif`

Translational diffusion object for the protein.

Type
IMP.atom.Diffusion

`Rot_diff`

Rotational diffusion object for the protein.

Type
IMP.atom.RigidBodyDiffusion

`Get_Hierarchy_From_PDB()`

Extracts the hierarchy of the protein from a PDB or CIF file.

This function reads a protein structure file (PDB or CIF) and extracts the molecular hierarchy and residue information for simulation.

Parameters
`None`

Returns
None

`combine_residues_to_fragment(`*Temp_Frag*`)`

Combines a set of residues into a single fragment.

This method calculates the center of mass, new radius, and total mass of a set of residues and groups them into a coarse-grained fragment.

Parameters
`Temp_Frag` (*list*) – List of residue data containing coordinates, radius, mass, and residue index.

Returns
None

`create_rigid_body_protein()`

Creates a rigid body representation of the protein.

This function sets up the protein as a rigid body for simulation, assigning its molecular hierarchy and defining its diffusion properties.

Parameters
`None`

Returns
None

create_rigid_body_protein_singlemodel()

Creates a rigid body representation of a single-model protein.

This function loads the protein structure from a PDB or CIF file, sets up its hierarchy, and defines a rigid body for coarse-grained simulation.

Parameters
None

Returns
None

setup_residues(*mol_id*, *residues*)

Sets up and configures residues for the protein.

This method initializes residues, assigns attributes such as mass and diffusion, and groups them into coarse-grained fragments based on the defined resolution.

Parameters

- **mol_id** (*str*) – Molecular identifier of the protein.
- **residues** (*list*) – List of residue particles to be set up.

Returns
None

translate_protein_to_center(*center=[0, 0, 0]*)

Translates the protein structure to a specified center position.

This function calculates the center of mass of the protein and applies a transformation to shift the protein to the desired location.

Parameters
center (*list, optional*) – The target center coordinates. Default is [0, 0, 0].

Returns
None

1.1.4 src.Simulation_Class module

Author: Neelesh Soni, neelesh@salilab.org, neeleshsoni03@gmail.com Date: April 5, 2024

src.Simulation_Class.**logger**

Description

Type
TYPE

class src.Simulation_Class.**Simulation**(*param*, *system*)

Bases: object

A class to represent and execute Brownian dynamics simulations on a protein system.

The *Simulation* class is responsible for setting up and running molecular simulations using Brownian dynamics. It initializes the simulation environment, defines system constraints, configures scoring functions, and records simulation trajectories.

param

Dictionary containing simulation parameters.

Type
dict

`system`

The protein system being simulated.

Type
System

`model`

The IMP model used for the simulation.

Type
IMP.Model

`simulation_time`

Total duration of the simulation in seconds.

Type
float

`temperature`

Temperature of the simulation system.

Type
float

`output_dir`

Directory where simulation output files are stored.

Type
str

`maximum_move`

Maximum displacement allowed per simulation step.

Type
float

`bd_step_size_sec`

Step size for Brownian dynamics in seconds.

Type
float

`output_traj_file`

Name of the trajectory output file.

Type
str

`bd`

Brownian dynamics optimizer used for the simulation.

Type
IMP.atom.BrownianDynamics

`scoring_function`

Scoring function used to evaluate simulation restraints.

Type
IMP.core.RestraintsScoringFunction

`sim_time_ns`

Total simulation time in nanoseconds.

Type
float

`sim_time_frames`

Number of simulation frames corresponding to the total simulation time.

Type
int

`rmf_dump_interval_frames`

Number of frames after which RMF trajectory snapshots are saved.

Type
int

`convert_time_ns_to_frames`(*time_ns*, *step_size_fs*)

Converts simulation time from nanoseconds to frames.

This function calculates the number of frames required based on the specified simulation time and Brownian dynamics step size.

Parameters

- **time_ns** (*float*) – Time in nanoseconds.
- **step_size_fs** (*float*) – Step size in femtoseconds.

Returns
Number of simulation frames.

Return type
int

`run`()

Executes the simulation.

This function applies necessary updates to the system and runs the Brownian dynamics optimizer for the specified number of simulation frames. It logs the scoring function before and after the simulation.

Parameters
None

Returns
None

`setup_brownian_dynamics`()

Configures the Brownian dynamics simulation settings.

This function sets up simulation time parameters, scoring functions, and RMF file configurations. It also defines constraints such as the maximum movement and temperature.

Parameters
None

Returns
None

1.1.5 src.System_Class module

Author: Neelesh Soni Contact: neelesh@salilab.org, neeleshsoni03@gmail.com Date: Feb 5, 2024

Description: This module defines the *System* class, which represents the simulation system, including proteins, interactions, and spatial constraints such as bounding boxes and periodic boundary conditions (PBCs). The *System* class integrates various IMP (Integrative Modeling Platform) components to facilitate molecular simulations.

Classes:

- System: Represents the molecular simulation system.

src.System_Class.**logger**

Logger instance for logging system events.

Type
logging.Logger

class src.System_Class.**System**(*param*)

Bases: object

A class to represent the simulation system, including proteins, interactions, and spatial constraints.

This class provides methods for setting up molecular simulations using the Integrative Modeling Platform (IMP). It defines proteins, interactions, and spatial boundaries within the system while applying restraints such as excluded volume, Lennard-Jones potentials, and membrane exclusion constraints.

bb

Outer bounding box for the simulation.

Type
IMP.algebra.BoundingBox3D

bb_cyt_bbss

Bounding box for the cytoplasm.

Type
IMP.core.BoundingBox3DSingletonScore

bb_cyt_harmonic

Harmonic upper bound for the cytoplasm.

Type
IMP.core.HarmonicUpperBound

bb_cytoplasm

Bounding box for the cytoplasm.

Type
IMP.algebra.BoundingBox3D

bb_harmonic

Harmonic upper bound for the simulation box.

Type
IMP.core.HarmonicUpperBound

bb_nuc_bbss

Bounding box for the nucleus.

Type
IMP.core.BoundingBox3DSingletonScore

`bb_nuc_harmonic`

Harmonic upper bound for the nucleus.

Type
IMP.core.HarmonicUpperBound

`bb_nucleus`

Bounding box for the nucleus.

Type
IMP.algebra.BoundingBox3D

`h_root`

Root hierarchy of the system.

Type
IMP.atom.Hierarchy

`interactions`

List of interactions between proteins in the system.

Type
list

`K_BB`

Harmonic upper bound constant for bounding boxes.

Type
float

`L`

Length of the bounding box sides.

Type
int

`mem_thickness`

Thickness of the membrane.

Type
int

`model`

The IMP model for simulations.

Type
IMP.Model

`outer_bbss`

Outer bounding box singleton score.

Type
IMP.core.BoundingBox3DSingletonScore

`proteins`

List of proteins included in the system.

Type
list

`restraints`

List of applied restraints, including spatial and interaction constraints.

Type
list

`state`

Current state of the system.

Type
IMP.pmi.topology.State

`sys`

IMP system topology object.

Type
IMP.pmi.topology.System

`tor_R`

Major radius for the toroidal membrane.

Type
float

`tor_r`

Minor radius for the toroidal membrane.

Type
float

`Iterate_Hierarchy()`

Iterates through the hierarchical structure of the system.

This function traverses the hierarchy of molecular components and logs information about the system structure.

Parameters
`None`

Returns
None

`add_LJ_potential_restraint(`*param*`)`

Adds a Lennard-Jones (LJ) potential restraint to the system.

This function applies an LJ potential to model attractive and repulsive interactions between proteins in the system.

Parameters
`param` (*dict*) – Dictionary containing LJ potential parameters, including: - min_dist (float): Minimum distance for interaction. - max_dist (float): Maximum distance for interaction. - epsilon (float): Energy well depth. - attractive_weight (float): Weight for attractive forces. - repulsive_weight (float): Weight for repulsive forces. - search_dist_cutoff (float): Cutoff distance for interaction search.

Returns
None

`add_components()`

Adds protein components to the simulation based on the provided system parameters. Proteins are created with specified copies, resolution, color, and diffusion coefficients.

Parameters
None

Returns
A list of Protein objects added to the system.

Return type
list

add_excluded_volume_restraint()

Adds an excluded volume restraint to prevent overlapping of proteins.

This function ensures that proteins are treated as spheres and prevents them from overlapping by applying an excluded volume restraint using their constituent residues.

Parameters
None

Returns
None

add_interaction(*protein1*, *protein2*, *interaction_type*, *strength*)

Creates and adds an interaction between two proteins in the system.

Parameters

- **protein1** (Protein) – The first protein involved in the interaction.
- **protein2** (Protein) – The second protein involved in the interaction.
- **interaction_type** (*str*) – Type of interaction (e.g., binding, electrostatic).
- **strength** (*float*) – Strength of the interaction.

Returns
None

add_membrane_exclusion_restraint(*NGH_proteins*)

Adds a membrane exclusion restraint to prevent proteins from entering a defined membrane region.

Parameters
NGH_proteins (*list*) – List of proteins near the membrane.

Returns
None

add_membrane_restraint(*NGH_proteins*)

Adds a membrane restraint to proteins using a custom implementation.

This function applies a constraint that ensures proteins remain in a specific membrane-associated region.

Parameters
NGH_proteins (*list*) – List of proteins near the membrane.

Returns
None

add_membrane_restraint2(*NGH_proteins*)

Adds a membrane exclusion restraint using predefined NPC (Nuclear Pore Complex) methods.

This function applies a membrane restraint on selected proteins using a predefined membrane exclusion model in IMP.

Parameters
NGH_proteins (*list*) – List of proteins near the membrane.

Returns
None

add_protein(*name*, *center*, *radius*, *mass*, *color*, *diffcoff=None*)

Adds a protein to the system with specified physical properties.

Parameters

- **name** (*str*) – Name of the protein.
- **center** (*list*) – Coordinates of the protein's center in 3D space.
- **radius** (*int*) – Radius of the protein.
- **mass** (*int*) – Mass of the protein.
- **color** (*int*) – Color index for display purposes.
- **diffcoff** (*float, optional*) – Diffusion coefficient of the protein. Defaults to None.

Returns
None

add_protein_from_structure(*name*, *pdbfile*, *pdb_multimodel*, *resolution*, *color*, *centerize*, *diffcoff=None*, *rot_diffcoff_scale=None*)

Adds a protein to the system using a structure file.

Parameters

- **name** (*str*) – Name of the protein.
- **pdbfile** (*str*) – Path to the PDB file containing the protein structure.
- **pdb_multimodel** (*bool*) – Whether the PDB file contains multiple models.
- **resolution** (*int*) – Resolution setting for the protein structure.
- **color** (*int*) – Color index for display purposes.
- **centerize** (*bool*) – Whether to center the protein coordinates.
- **diffcoff** (*float, optional*) – Diffusion coefficient of the protein. Defaults to None.
- **rot_diffcoff_scale** (*float, optional*) – Scaling factor for rotational diffusion. Defaults to None.

Returns
The added protein structure instance.

Return type
ProteinStructure

add_restraint(*restraint*)

Adds a restraint to the system to constrain molecular motion or interactions.

Parameters
restraint (*IMP.core.Restraint*) – The restraint object to be added.

Returns
None

apply_boundary_conditions()

Applies periodic or spatial boundary conditions to the system by constraining protein positions within predefined bounding regions.

Parameters
None

Returns
None

apply_cytoplasm_box_boundary_conditions(*proteins*)

Applies bounding box constraints to the proteins inside the cytoplasm.

This function ensures that proteins within the cytoplasm remain within the defined cytoplasmic bounding box using singleton restraints.

Parameters
proteins (*list*) – List of Protein objects representing proteins in the cytoplasm.

Returns
None

apply_cytoplasm_sphere_boundary_conditions(*proteins*)

Applies spherical boundary conditions to the cytoplasm.

This function ensures that proteins inside the cytoplasm remain within the defined cytoplasmic sphere boundary using singleton restraints.

Parameters
proteins (*list*) – List of Protein objects representing proteins in the cytoplasm.

Returns
None

apply_nucleoplasm_box_boundary_conditions(*proteins*)

Applies bounding box constraints to the proteins inside the nucleoplasm.

This function ensures that proteins within the nucleoplasm stay confined within the defined nucleoplasmic bounding box using singleton restraints.

Parameters
proteins (*list*) – List of Protein objects representing proteins in the nucleoplasm.

Returns
None

apply_nucleoplasm_excluded_boundary_conditions(*proteins*)

Applies exclusion boundary conditions to the nucleoplasm.

This function ensures that proteins are excluded from the nucleoplasm by enforcing a minimum distance constraint from the nucleoplasmic sphere.

Parameters
proteins (*list*) – List of Protein objects representing proteins to be excluded from the nucleoplasm.

Returns
None

`apply_nucleoplasm_sphere_boundary_conditions`(*proteins*)

Applies spherical boundary conditions to the nucleoplasm.

This function ensures that proteins inside the nucleoplasm remain within the defined spherical boundary using singleton restraints.

Parameters
`proteins` (*list*) – List of Protein objects representing proteins in the nucleoplasm.

Returns
None

`create_bounding_box_and_pbc`()

Defines the bounding boxes and periodic boundary conditions (PBCs) for the simulation space.

Parameters
`None`

Returns
None

`create_cell`()

Creates a spherical representation of the cell with mass and color attributes.

Parameters
`None`

Returns
The created particle representing the cell.

Return type
IMP.Particle

`create_cell_and_nucleus_bounding_sphere`()

Creates bounding spheres for the cytoplasm and nucleus, defining the spatial constraints of the system.

Parameters
`None`

Returns
None

`create_cell_bounding_sphere`()

Creates a bounding sphere for the cytoplasm to define spatial constraints for cellular components.

Parameters
`None`

Returns
None

`create_nucleus`()

Creates a spherical representation of the nucleus with mass and color attributes.

Parameters
`None`

Returns
The created particle representing the nucleus.

Return type
IMP.Particle

shuffle_neighborhood_proteins(*bounding_box*, *NGH_proteins*)

Randomly shuffles the positions of proteins within a specified bounding box.

Parameters

- **bounding_box** (*str*) – Type of bounding box ('nucleoplasm', 'cytoplasm', or 'default').
- **NGH_proteins** (*list*) – List of proteins to shuffle.

Returns
None

shuffle_proteins(*bounding_box*, *Proteins*)

Randomly shuffles the positions of proteins within a specified bounding box.

Parameters

- **bounding_box** (*str*) – The bounding box ('nucleoplasm', 'cytoplasm', 'cell', or 'box').
- **Proteins** (*list*) – List of proteins to shuffle.

Returns
None

update()

Updates the interactions within the system.

This function iterates over all interactions in the system and applies the specified interaction forces or constraints.

Parameters
None

Returns
None

1.1.6 src.Analysis_Class module

class src.Analysis_Class.**Analysis**(*trajectory_file*)

Bases: object

A class for analyzing and visualizing protein trajectories from simulation data.

The *Analysis* class provides methods to read simulation trajectory files, extract protein movement data, convert trajectory data to JSON format, and generate visual representations of protein movement.

trajectory_file

Path to the RMF trajectory file.

Type
str

Protein_Dict

Dictionary containing protein trajectory data, where keys are protein names and values store radii and coordinate lists.

Type
OrderedDict

calculate_interaction_matrix(*trajectory_data*, *distance_threshold=30.0*, *selected_proteins=None*)

Computes an interaction matrix for protein-protein interactions based on their coordinates. Merges different copies of the same protein type (e.g., NUP85_0, NUP85_1 → NUP85).

Parameters

- **trajectory_data** (*dict*) – Dictionary containing protein trajectory data.
- **distance_threshold** (*float*) – Maximum distance (in Angstroms) for proteins to be considered interacting.
- **selected_proteins** (*list, optional*) – List of unique protein types (e.g., ['NUP85', 'ULP1']) to include. If None, all proteins are included.

Returns

Interaction matrix (NxN) where N is the number of unique protein types. list: List of unique protein names.

Return type

np.ndarray

calculate_interaction_matrix_no_grouping(*trajectory_data*, *distance_threshold=30.0*, *selected_proteins=None*)

Computes an interaction matrix for protein-protein interactions based on their coordinates.

Parameters

- **trajectory_data** (*dict*) – Dictionary containing protein trajectory data.
- **distance_threshold** (*float*) – Maximum distance (in Angstroms) for proteins to be considered interacting.
- **selected_proteins** (*list, optional*) – List of protein names to include in the interaction matrix. If None, all proteins are included.

Returns

Interaction matrix (NxN) where N is the number of selected proteins. list: List of selected protein names.

Return type

np.ndarray

calculate_msd(*protein_trajectory*)

Calculates the Mean Square Displacement (MSD) for a given protein trajectory.

Parameters

protein_trajectory (*np.ndarray*) – Nx3 array containing (x, y, z) coordinates over time.

Returns

MSD values for different time lags.

Return type

np.ndarray

calculate_rdf(*trajectory_data*, *selected_proteins*, *bin_size=5.0*, *max_distance=100.0*)

Computes the Radial Distribution Function (RDF) for selected proteins.

Parameters

- **trajectory_data** (*dict*) – Dictionary containing protein trajectory data.
- **selected_proteins** (*list*) – List of proteins for which RDF is computed.
- **bin_size** (*float*) – Bin width for RDF computation (in Angstroms).

- **max_distance** (*float*) – Maximum distance to consider (in Angstroms).

Returns
Distance bins. np.ndarray: RDF values.

Return type
np.ndarray

convert_trajectory_to_json(*oname*, *precision=2*)

Converts the trajectory data into a JSON file.

Parameters

- **oname** (*str*) – Name of the output JSON file.
- **precision** (*int, optional*) – Decimal precision for coordinates. Default is 2.

Returns
None

normalize_interaction_matrix(*interaction_matrix*, *method='global'*)

Normalizes the interaction matrix into probabilities.

Parameters

- **interaction_matrix** (*np.ndarray*) – Raw interaction frequency matrix.
- **method** (*str*) – Normalization method. "row" (default) → Row-wise normalization (conditional probabilities). "global" → Global normalization (all probabilities sum to 1).

Returns
Normalized probability matrix.

Return type
np.ndarray

plot_interaction_matrix(*interaction_matrix*, *protein_names*, *save_path=None*, *title='Interaction Matrix'*)

Plots a heatmap of the normalized interaction matrix.

Parameters

- **interaction_matrix** (*np.ndarray*) – Normalized interaction matrix.
- **protein_names** (*list*) – List of protein names for axis labels.
- **save_path** (*str, optional*) – If provided, saves the plot instead of displaying.

Returns
None

plot_msd(*msd_values*, *save_path=None*)

Plots the Mean Square Displacement (MSD) over time.

Parameters

- **msd_values** (*np.ndarray*) – Computed MSD values.
- **save_path** (*str, optional*) – If provided, saves the plot instead of displaying.

Returns
None

plot_protein_trajectory(*protein_name*, *trajectory*, *save_path=None*)

Plots the 3D trajectory of a single protein.

Parameters

- **protein_name** (*str*) – Name of the protein.
- **trajectory** (*list*) – List of (x, y, z) coordinates for the protein trajectory.
- **save_path** (*str, optional*) – Path to save the plot. If None, the plot is displayed.

Returns
None

plot_rdf(*r_bins*, *rdf_values*, *save_path=None*)

Plots the Radial Distribution Function (RDF).

Parameters

- **r_bins** (*np.ndarray*) – Distance bins.
- **rdf_values** (*np.ndarray*) – Computed RDF values.
- **save_path** (*str, optional*) – If provided, saves the plot instead of displaying.

Returns
None

plot_two_protein_trajectory(*protein_name1*, *trajectory1*, *protein_name2*, *trajectory2*, *save_path=None*)

Plots the 3D trajectories of two proteins.

Parameters

- **protein_name1** (*str*) – Name of the first protein.
- **trajectory1** (*list*) – List of (x, y, z) coordinates for the first protein.
- **protein_name2** (*str*) – Name of the second protein.
- **trajectory2** (*list*) – List of (x, y, z) coordinates for the second protein.
- **save_path** (*str, optional*) – Path to save the plot. If None, the plot is displayed.

Returns
None

print_json(*oname*)

Loads and prints the JSON trajectory data.

Parameters
oname (*str*) – Path to the JSON file.

Returns
None

read_trajectory()

Reads the simulation trajectory from an RMF file.

This function extracts protein coordinates at different frames and stores them in *Protein_Dict*.

Returns

A dictionary containing protein names as keys and a list of
their radius and trajectory coordinates as values.

Return type
OrderedDict

1.1.7 src.Input_Parser module

class src.Input_Parser.**InputParser**(*file_path*)

Bases: object

A class for parsing and loading simulation parameters from a YAML configuration file.

The *InputParser* class reads simulation parameters from a specified YAML file and makes them available as a dictionary for use in the simulation setup.

parameters

A dictionary containing the parsed simulation parameters.

Type
dict

get_parameters()

Retrieves the simulation parameters.

This method returns the simulation parameters dictionary, specifically extracting the "simulation_parameters" section if available.

Returns
The dictionary containing simulation parameters.

Return type
dict

load_simulation_parameters(*file_path*)

Loads the simulation parameters from a YAML file.

This method reads the YAML file, parses its contents into a dictionary, and stores it in the *parameters* attribute.

Parameters
file_path (*str*) – Path to the YAML file containing simulation parameters.

Returns
Dictionary containing the simulation parameters.

Return type
dict

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

S

INDEX

D

G

H

I

K

L

M

N

O

P

R

S

T

U