hpsOpenMM Documentation

Release v1.3

hpsOpenMM

November 06, 2022

CONTENTS:

This documentation is currently being generated as we finalize

CHAPTER

ONE

INTRODUCTION

Hydropathy Scale are the model of protein systems based on simplifications made over classical Molecular Dynamics (MD) force fields.

The hps model is a Python library that offers flexibility to set up coarse-grained simulation of IDP using the MD framework of OpenMM toolkit. The codebase is based on sbmOpenMM scripts. It automates the creation of `openmm.system` classes that contain the necessary force field parameters to run molecular dynamics simulations using a protein structure as the only necessary inputs.

hps is divided in three main classes:

1. `geometry`
2. `models`
3. `system`

The first class, `geometry`, contains methods to calculate the geometrical parameters from the input structures. It's not useful in current need of simulation method.

The second class, `models`, allows to easily set up CG models.

The third class, `system`, is the main class that holds all the methods to define, modify and create CG system to be simulated with OpenMM.

The library is open-source and offers flexibility to simulate IDPs.

CHAPTER

TWO

SIMULATION CONTROL OPTIONS

This file contains an example of how config file of simulation looks like.

```
[OPTIONS]
md_steps = 30_000 # number of steps
dt = 0.01 ; time step in ps
nstxout = 1000 ; number of steps to write checkpoint = nstxout
nstlog = 1000 ; number of steps to print log
nstcomm = 100 ; frequency for center of mass motion removal
; select HPS model, available options: hps_kr, hps_urry, or hps_ss
model = hps_urry

; control temperature coupling
tcoupl = yes
ref_t = 310 ; Kelvin- reference temperature
tau_t = 0.01 ; ps^-1

;pressure coupling
pcoupl = yes
ref_p = 1
frequency_p = 25

; Periodic boundary condition: if pcoupl is yes then pbc must be yes.
pbc = yes
; if pbc=yes, then use box_dimension option to specify box_dimension = x or [x, y, z],␣
↪unit of nanometer
box_dimension = 30 ; [30, 30, 60]

; input
protein_code = FUS_100chains
pdb_file = FUS_100chains.pdb
; output
checkpoint = FUS_100chains.chk
;Use GPU/CPU
device = GPU
; If CPU is specified, then use ppn variable
ppn = 4
;Restart simulation
restart = no
minimize = yes ;if not restart, then minimize will be loaded, otherwise, minimize=False
```

2.1 General information

Simulation parameters are input from *.ini* file which is loaded by *ConfigParser* module in Python. The section title `[OPTIONS]` is required, do not change section's name.

- Comment can be inline or in new line, start with *;* or *#*
- Keyword and value can be separated by = or *:*

2.2 Run control

```
md_steps:   (long int)
            (1) Maximum number of steps to integrate or minimize
------------------------------------------------------------------------------
dt:         (double)
            (0.01)[ps] Time step for integration
------------------------------------------------------------------------------
nstxout:    (int)
            (1) [step] number of steps that elapse between writing coordinates to output␣
↪trajectory file,
                  the last coordinates are always written
------------------------------------------------------------------------------
nstlog:     (int)
            (1) number of steps that elapse between writing energies to the log file
------------------------------------------------------------------------------
nstcomm:    (int)
            (100) frequency for center of mass motion removal
```

2.3 Model parameter

There are three models supported now: *hps_kr*, *hps_urry* and *hps_ss*.

hps_kr has parameters for a wide range of residues, i.e RNA, phosphorylation residues . . . but this model is less accurate

```
model:      (string)
            hps_kr: Kapcha-Rossy hydropathy scale, parameterize from OPLS-AA forcefield

            hps_urry (default): Urry hydropathy scale, parameterize from experiment

            hps_ss: hps_urry with bonded potential (angle and torsion)
```

2.4 Temperature coupling

```
tcoupl:     (bool)
            yes (default) : The only available option for now, we don't care about NVE␣
↪ensemble.
-------------------------------------------------------------------------------------
ref_t:      (double)
            (300) [K] : Reference temperature in unit of Kelvin
-------------------------------------------------------------------------------------
tau_t:      (double)
            [ps^-1] : The friction coefficient which couples the system to the heat bath␣
↪(in inverse picoseconds)
```

2.5 Pressure coupling

```
pcoupl      (bool)
            yes : Using pressure coupling

            no (default) : Run on NVT ensemble only
-------------------------------------------------------------------------------------
ref_p       (double)
             (1) [bar] The default pressure acting on the system.
-------------------------------------------------------------------------------------
frequency_p (int)
            (25) [steps] the frequency at which Monte Carlo pressure changes should be␣
↪attempted
```

2.6 Periodic boundary condition:

if pcoupl is yes then pbc must be yes.

```
pbc         (bool)
            yes : Using periodic boundary condition.
                    If this option is chosen, then it will affect to non-bonded forces␣
↪in the system,
                    and the coordinate writen in PDB and DCD file as well. No worries␣
↪since I have handled these.

            no (default) : Without periodic boundary condition.
-------------------------------------------------------------------------------------
box_dimension   (float or list of float)
            [nm] An example of box dimension:
            If you want a cubic box of 30x30x30 nm^3, put: 30 or [30, 30, 30]
            If you want a rectangular box? Put:  [30, 30, 60]
```

2.7 File input/output

```
protein_code    (string)
                String for output prefix, i.e {protein_code}.dcd, {protein_code}.log
--------------------------------------------------------------------------------
pdb_file        (string)
                [.pdb, .cif] Input structure for loading topology and initial coordinate
--------------------------------------------------------------------------------
checkpoint      (string)
                [.chk] Checkpoint file name, here I ask you to provide it explicitly␣
↪since
                       because checkpoint can be used to load state or save state.
                       in case if you restart simulation with different name, you have␣
↪to provide it.
```

2.8 Simulation platform

Simulation can be run on CPU with number of threads is control by *ppn* or using GPU. If *device=CPU* then ppn need to be specify, otherwise simulation will run on 1 core

```
device          (string)
                GPU : Use gpu to run simulation

                CPU (default) : use cpu to run simulation, if you specify cpu, you␣
↪should modify ppn option, it control
                       how many cores will be used to run simulation, if not, default␣
↪is 1.
--------------------------------------------------------------------------------
ppn             (int)
                (1) [threads] Number of threads used to run simulation on CPU. When␣
↪using GPU,
                           performance is boosted a lot so ppn in that case is set to 1.
```

2.9 Restart simulation

```
restart         (bool)
                yes : restart simulation from checkpoint file. This can be True, 1 or␣
↪whatever are not (FALSE)
                       in python condition. If this option is selected, minimize will␣
↪be force to False.

                no (default) : Run simulation from beginning, if this option is selected,
↪ you can choose if you want to minimize your
                    system before running simulation.
--------------------------------------------------------------------------------

minimize        (bool)
                yes (default) : perform energy minimization before run molecular␣
```

(continues on next page)

(continued from previous page)

```
↪dynamics.

               no : Not running energy minimization. This is default option when␣
↪restart option is set to yes.
```

CHAPTER

THREE

PARAMETERS

Dictionary contains parameters for hps model. First level is the model name

- HPS-Kr scale was taken from:

Dignon, G. L., Zheng, W., Kim, Y. C., Best, R. B., ; Mittal, J. (2018). Sequence determinants of protein phase behavior from a coarse-grained model. PLoS Computational Biology, 1–23. https://doi.org/10.1101/238170

- Parameter for Nucleic acids (KR scale):

Regy, R. M., Dignon, G. L., Zheng, W., Kim, Y. C., Mittal, J. (2020). Sequence dependent phase separation of protein-polynucleotide mixtures elucidated using molecular simulations. Nucleic Acids Research, 48(22), 12593–12603. https://doi.org/10.1093/nar/gkaa1099

- Phosphorylation version of some residues for KR scale are taken from:

Perdikari, T. M., Jovic, N., Dignon, G. L., Kim, Y. C., Fawzi, N. L., Mittal, J. (2021). A predictive coarse-grained model for position-specific effects of post-translational modifications. Biophysical Journal, 120(7), 1187–1197. https://doi.org/10.1016/j.bpj.2021.01.034

3.1 Note on hps (lambda) in urry scale:

These parameters were shifted by 0.08 from original parameters directly.

in the original paper: Regy, R. M., Thompson, J., Kim, Y. C., ; Mittal, J. (2021). Improved coarse-grained model for studying sequence dependent phase separation of disordered proteins. Protein Science, 30(7), 1371–1379. https://doi.org/10.1002/pro.4094

..math::

lambda_{ij} = muy lambda_0_ij - delta

muy=1, delta= 0.08 is the optimal set for the set of 42 proteins they studied. lambda_{ij} = lambda0_{ij} - 0.08 = 0.5*(lambda_i+lambda_j) - 0.08 = 0.5(lambda_i -0.08 + lambda_j-0.08)

In both version, KR and Urry, we can tune directly lambda parameter in Urry by 0.08 so we can use only one equation for two model (choose parameter when passing hps_scale parameter)

`hps.parameters.model_parameters.`**`parameters`**

dictionary contains model parameters.

CHAPTER

FOUR

SYSTEM

A class containing methods and parameters for generating CG systems to be simulated using the OpenMM interface.

It offers flexibility to create default and custom CG systems and to easily modify their parameters.

class hps.core.**system**(*structure_path*, *hps_scale*)

A class containing methods and parameters for generating CG systems to be simulated using the OpenMM interface. It offers flexibility to create default and custom CG systems and to easily modify their parameters.

Parameters

- **structure_path** (*string [requires]*) – Name of the input PDB or CIF file
- **hps_scale** (*'hps_kr','hps_urry' [optional, default='hps_urry']*) – Hydropathy scale. Currently, there are two models are supported.

structure

Object that holds the information of OpenMM PDB or CIF parsing methods.

Type
openmm.app.pdbfile.PDBFile or openmm.app.pdbxfile.PDBxFile

topology

OpenMM topology of the model.

Type
openmm.app.topology.Topology

positions

Atomic positions of the model.

Type
unit.quantity.Quantity

particles_mass

Mass of each particle. If float then uniform masses are given to all particles. If list per-particle masses are assigned.

Type
float or list

particles_charge

Charge of each particle.

Type
list

rf_sigma

Sigma parameter used in the pairwise force object. This is vdw Radius of beads

Type
`float`

atoms

A list of the current atoms in the model. The items are `openmm.app.topology.atoms` initialised classes.

Type
`list`

n_atoms

Total numer of atoms in the model.

Type
`int`

bonds

A dict that uses bonds (2-tuple of `openmm.app.topology.bonds` objects) present in the model as keys and their forcefield properties as values.

Type
`collections.OrderedDict`

bonds_indexes

A list containing the zero-based indexes of the atoms defining the bonds in the model.

Type
`list`

n_bonds

Total number of bonds in the model.

Type
`int`

bonded_exclusions_index

Exclusion rule for nonbonded force. =1 for hps_kr and hps_urry, =3 for hps_ss

Type
`int`

harmonicBondForce

Stores the OpenMM `HarmonicBondForce` initialised-class. Implements a harmonic bond potential between pairs of particles, that depends quadratically on their distance.

Type
`openmm.HarmonicBondForce`

n_angles

Total number of angles in the model.

Type
`int`

gaussianAngleForce

Stores the OpenMM `CustomAngleForce` initialised-class. Implements a Gaussian angle bond potential between pairs of three particles.

Type
openmm.CustomAngleForce

n_torsions

Total number of torsion angles in the model.

Type
int

gaussianTorsionForce

Stores the OpenMM CustomTorsionForce initialised-class. Implements a Gaussian torsion angle bond potential between pairs of four particles.

Type
openmm.CustomTorsionForce

yukawaForce

Stores the OpenMM CustomNonbondedForce initialized-class. Implements the Debye-Huckle potential.

Type
openmm.CustomNonbondedForce

ashbaugh_HatchForce

Stores the OpenMM CustomNonbondedForce initialized-class. Implements the pairwise short-range potential.

Type
openmm.CustomNonbondedForce

forceGroups

A dict that uses force names as keys and their corresponding force as values.

Type
collections.OrderedDict

system

Stores the OpenMM System initialised class. It stores all the forcefield information for the hps model.

Type
openmm.System

loadForcefieldFromFile()

Loads forcefield parameters from a force field file written with the dumpForceFieldData() method.

__init__(*structure_path*, *hps_scale*)

Initialises the hps OpenMM system class.

Parameters

- **structure_path** (*string [requires]*) – Name of the input PDB or CIF file
- **hps_scale** (*'hps_kr','hps_urry', or 'hps_ss' [optional, default='hps_urry']*) – Hydropathy scale. Currently, there are three models are supported.

Return type
None

getAtoms()

Reads atoms from topology, adds them to the main class and sorts them into a dictionary to store their forcefield properties.

After getCAlphaOnly, C-alpha atoms are stored on `self.topology only`. We need to add them to atoms attribute and system also. Adds `atoms` in the `OpenMM topology` instance to the `hpsOpenMM system` class.

Return type
None

`getBonds`(*except_chains=None*)

Reads bonds from topology, adds them to the main class and sorts them into a dictionary to store their forcefield properties.

Adds `bonds` in the `OpenMM topology` instance to the `hpsOpenMM system` class.

Parameters
except_chains (*String [optional]*) –

Return type
None

`setBondForceConstants`(*bond_force_constant*)

Change the forcefield parameters for bonded terms.

Set the harmonic bond constant force parameters. The input can be a float, to set the same parameter for all force interactions, or a list, to define a unique parameter for each force interaction.

Parameters
bond_force_constant (*float or list*) – Parameter(s) to set up for the harmonic bond forces.

Return type
None

`setParticlesRadii`(*particles_radii*)

Change the excluded volume radius parameter for each atom in the system.

Set the radii of the particles in the system. The input can be a float, to set the same radius for all particles, or a list, to define a unique radius for each particle.

Parameters
particles_radii (*float or list*) – Radii values to add for the particles in the hpsOpenMM system class.

Return type
None

`setParticlesCharge`(*particles_charge*)

Set the charge of the particles in the system. The input can be a float, to set the same charge for all particles, or a list, to define a unique charge for each particle.

Parameters
particles_charge (*float or list*) – Charge values to add for the particles in the hpsOpenMM system class.

Return type
None

`setParticlesHPS`(*particles_hps*)

Set the hydropathy scale of the particles in the system. The input can be a float, to set the same hydropathy for all particles, or a list, to define a unique hydropathy for each particle.

Parameters
particles_hps (*float or list*) – HPS scale values to add for the particles in the hpsOpenMM system class.

Return type
None

addYukawaForces(*use_pbc: bool*) → None

Creates a nonbonded force term for electrostatic interaction DH potential.

Creates an `openmm.CustomNonbondedForce()` object with the parameters sigma and epsilon given to this method. The custom non-bonded force is initialized with the formula:

$$energy = f \times \frac{q_1 q_2}{\epsilon_r \times r} \times e^{(-r/lD)}$$

where $f = \frac{1}{4\pi\epsilon_0} = 138.935458$ is the factor for short to convert dimensionless in calculation to $kj.nm/(mol \times e^2)$ unit.

$\epsilon_r = 80$: Dielectric constant of water at 100mM mono-valent ion

The force object is stored at the `yukawaForce` attribute.

Parameters
use_pbc (*(bool) whether use PBC, cutoff periodic boundary condition*) –

Return type
None

addAshbaughHatchForces(*use_pbc: bool*) → None

Creates a nonbonded force term for pairwise interaction (customize LJ 12-6 potential).

Creates an `openmm.CustomNonbondedForce()` object with the parameters sigma and epsilon given to this method. The custom non-bonded force is initialized with the formula: (note: hps here is λ_{ij}^0 in the paper)

Unlike `BondForce` class, where we specify index for atoms pair to add bond, it means that number of bondForces may differ from number of particle. `NonBondedForce` is added to all particles, hence we don't need to pass the `atom index`.

$$\Phi_{i,j}^{vdw}(r) = step(2^{1/6}\sigma_{ij} - r) \times \left(4\epsilon\left[\left(\frac{\sigma_{ij}}{r}\right)^{12} - \left(\frac{\sigma_{ij}}{r}\right)^{6}\right] + (1-\lambda_{ij})\epsilon\right) + \left[1 - step(2^{1/6}\sigma_{ij} - r)\right] \times \left[(\lambda_{ij}) \times 4\epsilon\left[\left(\frac{\sigma_{ij}}{r}\right)^{12} - \left(\frac{\sigma_{ij}}{r}\right)^{6}\right]\right]$$

Here, $\sigma = \frac{(\sigma_1+\sigma_2)}{2}$; $\lambda_{ij}^0 = \frac{(\lambda_i+\lambda_j)}{2}$; $\epsilon = 0.8368kj/mol$

The force object is stored at the `ashbaugh_HatchForce` attribute.

epsilon
[float] Value of the epsilon constant in the energy function.

sigma
[float or list] Value of the sigma constant (in nm) in the energy function. If float the same sigma value is used for every particle. If list a unique parameter is given for each particle.

cutoff
[float] The cutoff distance (in nm) being used for the non-bonded interactions.

Parameters
use_pbc (*bool. Whether use PBC, cutoff periodic boundary condition*) –

Return type
None

createSystemObject(*check_bond_distances: bool = True*, *minimize: bool = False*, *check_large_forces: bool = True*, *force_threshold: float = 10.0*, *bond_threshold: float = 0.5*) → None

Creates OpenMM system object adding particles, masses and forces. It also groups the added forces into Force-Groups for the hpsReporter class.

Creates an `openmm.System()` object using the force field parameters given to the 'system' class. It adds particles, forces and creates a force group for each force object. Optionally the method can check for large bond distances (default) and minimize the atomic positions if large forces are found in any atom (default False).

Parameters

- **minimize** (*boolean (False)*) – Whether to minimize the system if large forces are found.
- **check_bond_distances** (*boolean (True)*) – Whether to check for large bond distances.
- **check_large_forces** (*boolean (False)*) – Whether to print force summary of force groups
- **force_threshold** (*float (10.0)*) – Threshold to check for large forces.
- **bond_threshold** (*float (0.5)*) – Threshold to check for large bond distances.

Return type
None

checkBondDistances(*threshold: float = 0.5*) → None

Searches for large bond distances for the atom pairs defined in the 'bonds' attribute. It raises an error when large bonds are found.

Parameters
threshold (*(float, default=0.5 nm)*) – Threshold to check for large bond distances.

Return type
None

checkLargeForces(*minimize: bool = False*, *threshold: float = 10*) → None

Prints the hps system energies of the input configuration of the system. It optionally checks for large forces acting upon all particles in the hps system and iteratively minimizes the system configuration until no forces larger than a threshold are found.

Parameters

- **threshold** (*(float, default=10)*) – Threshold to check for large forces.
- **minimize** (*(bool, default= False)*) – Whether to iteratively minimize the system until all forces are lower or equal to the threshold value.

Return type
None

addParticles() → None

Add particles to the system OpenMM class instance.

Add a particle to the system for each atom in it. The mass of each particle is set up with the values in the `particles_mass` attribute.

addSystemForces() → None

Add forces to the system OpenMM class instance. It also save names for the added forces to include them in the reporter class.

Adds generated forces to the system, also adding a force group to the `forceGroups` attribute dictionary.

dumpStructure(*output_file: str*) → None

Writes a structure file of the system in its current state.

Writes a PDB file containing the currently defined CG system atoms and its positions.

Parameters
output_file (*string*) – name of the PDB output file.

Return type
None

dumpTopology(*output_file: str*) → None

Writes a topology file of the system in PSF format, this is used for visualization and post-analysis.

Writes a file containing the current topology in the hpsOpenMM system. This file contains topology of system, used in visualization and analysis.

Here, we used `parmed` to load `openMM topology` and `openMM system` to create `Structure` object in `parmed`. Because parmed doesn't automatically recognize `charge`, `mass` of atoms by their name. We need to set `charge`, `mass` back to residues properties.

Parameters
output_file (*string [requires]*) – name of the output PSF file.

Return type
None

dumpForceFieldData(*output_file: str*) → None

Writes to a file the parameters of the forcefield.

Writes a file containing the current forcefield parameters in the CG system.

Parameters
output_file (*string [requires]*) – name of the output file.

Return type
None

setCAMassPerResidueType()

Sets alpha carbon atoms to their average residue mass. Used specially for modifying alpha-carbon (CA) coarse-grained models.

Sets the masses of the alpha carbon atoms to the average mass of its amino acid residue.

Return type
None

setCARadiusPerResidueType()

Sets alpha carbon atoms to their average residue mass. Used specially for modifying alpha-carbon (CA) coarse-grained models.

Sets the excluded volume radii of the alpha carbon atoms to characteristic radii of their corresponding amino acid residue.

Return type
None

`setCAChargePerResidueType()`

Sets the charge of the alpha carbon atoms to characteristic charge of their corresponding amino acid residue.

Return type
None

`setCAHPSPerResidueType()`

Sets alpha carbon atoms to their residue hydropathy scale. Used specially for modifying alpha-carbon (CA) coarse-grained models.

Sets the HPS model of the alpha carbon atoms using corresponding scale.

Return type
None

`static _setParameters(`*term*, *parameters*`)`

General function to set up or change force field parameters. protected method, can be called only inside class system.

Parameters

- **term** (*dict*) – Dictionary object containing the set of degrees of freedom (DOF) to set up attributes to (e.g. bonds attribute)
- **parameters** (*integer or float or list*) – Value(s) for the specific forcefield parameters. If integer or float, sets up the same value for all the DOF in terms. If a list is given, sets a unique parameter for each DOF.

Return type
None

CHAPTER

FIVE

MODELS

The models class contains three methods for automatic setting up predefined potentials.

It works by initializing a system class with the necessary force field parameters.

5.1 Coarse grained, alpha-carbon (CA), model

The coarse grained method represents the protein system as beads centered at the alpha carbons of each residue in the protein.

It uses harmonic potentials to hold the covalent connectivity and geometry of the beads.

Torsional geometries are modeled with a periodic torsion potential.

Native contacts are represented through the use of Lennard-Jones potentials that allow to form and break non-bonded interactions, permitting complete and local unfolding of the structures.

To create a CA model, call: `hps.models.getCAModel(pdb_file, hps_scale)`

Here, pdb_file is the path to the PDB format structure of the protein. hps_scale is hydropathy scale that are going to be used. `urry` or `kr`

The force field equations are:

$$H_A = \sum_{bonds} V_{bond} + \sum_{i,j} \Phi_{ij}^{vdw} + \sum_{i,j} \Phi_{i,j}^{el}$$

5.2 The Bonded potential:

$$V_{bond} = \frac{k_b}{2}(r - r_0)^2$$

Here the default values are $k_b = 8368kJ/(mol \times nm^2), r_0 = 0.382nm$

5.3 The Pairwise potential:

$$\Phi_{i,j}^{vdw}(r) = step(2^{1/6}\sigma_{ij} - r) \times \left(4\epsilon\left[\left(\frac{\sigma_{ij}}{r}\right)^{12} - \left(\frac{\sigma_{ij}}{r}\right)^{6}\right] + (1 - \mu \times \lambda_{ij}^{0} + \Delta) \times \epsilon\right)$$
$$+ \left[1 - step(2^{1/6}\sigma_{ij} - r)\right] \times \left[(\mu\lambda_{ij}^{0} - \Delta) \times 4\epsilon\left[\left(\frac{\sigma_{ij}}{r}\right)^{12} - \left(\frac{\sigma_{ij}}{r}\right)^{6}\right]\right]$$

Since the step function behaves like: `step(x) = 0 if x < 0,and =1 otherwise`, we can separate in multiple cases for short likes following:

$$\Phi_{i,j}^{vdw}(r) = 4\epsilon\left[\left(\frac{\sigma_{ij}}{r}\right)^{12} - \left(\frac{\sigma_{ij}}{r}\right)^{6}\right] + (1 - \mu \times \lambda_{ij}^{0} + \Delta) \times \epsilon, r \leq 2^{1/6}\sigma_{ij}$$
$$\Phi_{i,j}^{vdw}(r) = (\mu \times \lambda_{ij}^{0} - \Delta) \times \left(4\epsilon\left[\left(\frac{\sigma_{ij}}{r}\right)^{12} - \left(\frac{\sigma_{ij}}{r}\right)^{6}\right]\right), r > 2^{1/6}\sigma_{ij}$$

where, $\sigma_{i,j} = \frac{\sigma_i + \sigma_j}{2}$: is the vdW radius interaction of interacting beads

$\lambda_{ij}^{0} = \frac{\lambda_i + \lambda_j}{2}$: hydropathy scale interaction of residues

μ, Δ: are the only free parameters in the model. In Jeetain Mittal(2021) Protein Science, he simulated for 42 IDP proteins and fit Rg vs experimental values.

In the current implementation, hydropathy scales are taken from Urry model, $(\mu, \Delta) = (1, 0.08)$

Nonbonded exclusion rule is `1-2`, which we only exclude pair of atoms in bonded.

The cut-off distance for Lennard-Jone potential: $2.0nm$

5.4 The Debye-Huckle potential has following form:

$$\Phi_{ij}^{el}(r) = \frac{q_i q_j}{4\pi\epsilon_0 D r} e^{-\kappa r}$$

where, q_i, q_j are charge of residues i, j

ϵ_0: Vacuum permitivity. For convenient, we precalculated the electric conversion factor $\frac{1}{4\pi\epsilon_0} = 138.935485(9) kJ \times mol^1 \times nm \times e^2$.

D: dielectric constant, at 100mM mono-valence salt (NaCl), it takes values of 80. The dielectric constant here is fixed, but it can be temperature dependent as the function: $\frac{5321}{T} + 233.76 - 0.9297T + 0.1417 \times 10^{-2} \times T^2 - 0.8292 \times 10^{-6} \times T^3$

κ: inverse Debye length, at 100mM NaCl has values of $1nm^{-1}$

The cut-off distance for Electrostatics interactions: $3.5nm$

class hps.core.**models**

A class to hold functions for the automated generation of default hps models.

__init__()

static buildHPSModel(*structure_file: str*, *minimize: bool = False*, *hps_scale: str = 'hps_urry'*, *box_dimension: Optional[Any] = None*)

Creates an alpha-carbon only `hpsOpenMM` system class object with default initialized parameters.

Initializes a coarse-grained, carbon alpha (CA), hpsOpenMM system class from a structure and a contact file defining the native contacts for the coarse grained model.

The system creation steps are:

1) Add the geometrical parameters for the model.
2) Add the default force field parameters for the model.
3) Create the default force objects.
4) Create the OpenMM system class.

The method can be used to generate an initialized hpsOpenMM system class, that only contains the geometrical parameters, by passing the option default_parameters as False.

Parameters

- **structure_file** (*string [requires]*) – Path to the input structure file.
- **minimize** (*boolean (False)*) – If True the initial structure will undergo the energy minimization.
- **hps_scale** (*string [Optional, hps_urry]*) –

  HPS scale. There are three options correspond to two scale:

  - 'hps_urry': using Urry scale (default).
  - 'hps_ss': hps_urry with angle and torsion potential.
  - 'hps_kr': using Kapcha-Rossy scale.
- **box_dimension** (*float or array (None)*) – If box_dimension is supplied, then will use PBC. if float is given, then use cubic box if an array of (3,1) is given, then use rectangular box with the given dimension if not specify: do not use PBC

Returns

hps – Initialized hpsOpenMM.system class with default options for defining a coarse-grained CA force field.

Return type

hpsOpenMM.system

CHAPTER

SIX

DYNAMICS

Dynamics class contains two main functions: read config file and run simulation.

User only need to provide config file, e.g md.ini and specify parameters control simulation there.

class hps.dynamics.**Dynamics**(*config_file*)

Dynamics class contains two main functions: read config file and run simulation. User only need to provide config file, e.g md.ini and specify parameters control simulation there.

Parameters

config_file (*str*) – control parameters for simulation

md_steps

Number of steps to perform molecular dynamics simulation

Type

int [1, steps]

dt

time step for integration

Type

float [0.01, ps]

nstxout

number of steps that elapse between writing coordinates to output trajectory file, the last coordinates are always written

Type

int [1, steps]

nstlog

number of steps that elapse between writing energies to the log file, the last energies are always written

Type

int [1, steps]

nstcomm

frequency for center of mass motion removal

Type

int [100, steps]

model

Hydropathy scale

Type

str ['hps_urry']

`tcoupl`

Using temperature coupling.

Type
bool

`ref_t`

reference temperature for coupling

Type
float [Kelvin]

`tau_t`

ime constant for temperature coupling

Type
float [ps]

`pcoupl`

Pressure coupling

Type
bool

`ref_p`

The reference pressure for coupling.

Type
float [bar]

`frequency_p`

The frequency for coupling the pressure.

Type
int [25, steps]

`pbc`

Use periodic boundary conditions.

Type
bool

`box_dimension`

Box dimension defined the unit cell, better to use rectangular for simplicity

Type
float or list of float

`protein_code`

Prefix to write output file based on this parameter

Type
str

`checkpoint`

Checkpoint file name

Type
str

pdb_file

Input structure read to generate model.

Type
str

device

Device to perform simulation [GPU/CPU] if CPU is used, then need to provide number of threads to run simulation.

Type
str

ppn

In case simulation is run on CPU, use this parameter to control the number of threads to run simulation.

Type
int [1, cores]

restart

If simulation run from beginning or restart from checkpoint.

Type
bool [No]

minimize

If simulation run from beginning then need to perform energy minimization. If simulation restarted, this parameters will be override to False.

Type
bool

__init__(*config_file*)

read_config(*config_file*)

Read simulation control parameters from config file *.ini into class attributes.

TODO: check parameters in control file more carefully.
Raise error and exit immediately if something wrong.

CHAPTER

SEVEN

CHANGELOG

CHAPTER

EIGHT

ABOUT

PYTHON MODULE INDEX

h

INDEX

Symbols

A

B

C

D

F

G

H

L

M

N

P

R

S

T

Y