OpenSMOG

Release 1.0.4

Antonio B. Oliveira Jr., Vinícius G. Contessoto & Paul Whitford

Aug 21, 2021

GETTING STARTED

CHAPTER

ONE

INSTALLATION

1.1 Installing OpenSMOG

The **OpenSMOG** library can be installed via conda or pip, or compiled from source (GitHub).

1.1.1 Install via conda

The code below will install **OpenSMOG** from conda-forge.

```
conda install -c conda-forge OpenSMOG
```

1.1.2 Install via pip

The code below will install **OpenSMOG** from PyPI.

```
pip install OpenSMOG
```

1.1.3 Install OpenMM

The **OpenSMOG** library uses OpenMM API to run the molecular dynamics simulations. **OpenMM** may be installed from the conda-forge channel:

```
conda install -c conda-forge openmm
```

The following libraries are **required** when installing **OpenSMOG**:

- Python (>=3.6)
- NumPy (>=1.14)
- ElementTree XML (>=2.2.0)

1.2 Installing SMOG2

The input files for **OpenSMOG** simulations are generated using SMOG2 (version 2.4, or newer). Here, there is a quick installation guide based on conda (Linux and Windows-WSL only). Alternate installation options are described in the SMOG2 manual.

First, create a new environment and activate it:

```
conda create --name smog2.4 perl
```

```
conda activate smog2.4
```

Next, it is necessary to instal a few **Perl** modules (a complete list of modules is in the SMOG2 README file):

```
conda install -c bioconda perl-xml-simple perl-xml-libxml java-jdk
```

```
conda install -c eumetsat perl-pdl
```

```
perl -MCPAN -e 'install XML::Validator::Schema'
```

Enter the **Perl** and **smog2** paths in the configure.smog2 file.

Then load and test your **smog2** configuration:

```
source configure.smog2
```

```
./test-config
```

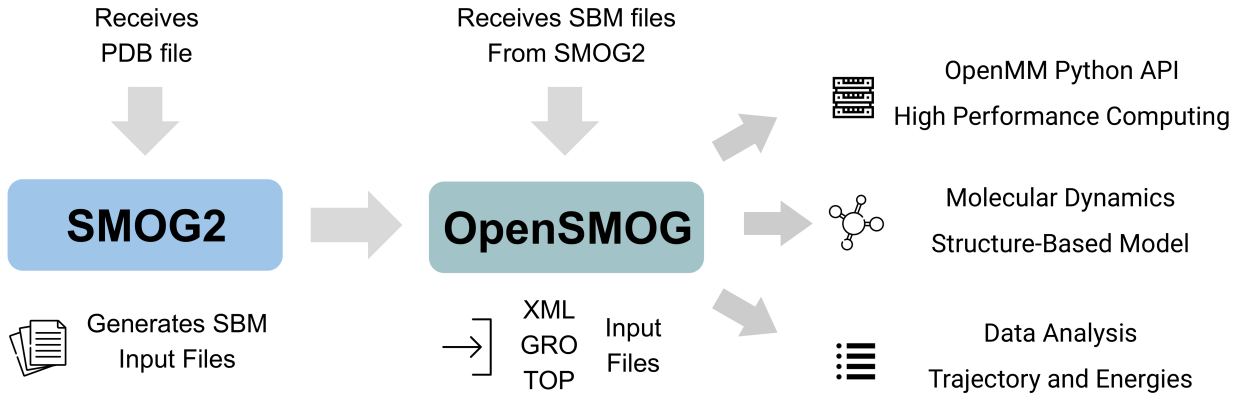
It is also **STRONGLY** recommended that you download **smog-check** (available at smog-server.org) and run all tests before using **smog2** for production calculations.

CHAPTER

TWO

INTRODUCTION

OpenSMOG is a Python library for performing molecular dynamics simulations using Structure-Based Models [[1]]. OpenSMOG uses the OpenMM Python API that supports a wide variety of potential forms, which includes the commonly employed C-alpha [[2]] and All-Atom [[3]] models. The input files are generated using the SMOG2 software package with the flag `-OpenSMOG`. Details on SMOG2 usage can be found in the SMOG2 User Manual.



CHAPTER

THREE

OPENSMOG

The OpenSMOG classes perform molecular dynamics using Structure-Based Models (SBM) for biomolecular simulations. OpenSMOG uses force fields generated by SMOG 2, and it allows the simulations of a wide variety of potential forms, including commonly employed C-alpha and all-atom variants. Details about the default models in SMOG 2 can be found in the following resources:

- **SMOG server**: https://smog-server.org/smog2/
- **C-alpha**: Clementi, C., Nymeyer, H. and Onuchic, J.N., 2000. Topological and energetic factors: what determines the structural details of the transition state ensemble and "en-route" intermediates for protein folding? An investigation for small globular proteins. Journal of molecular biology, 298(5), pp.937-953.
- **All-Atom**: Whitford, P.C., Noel, J.K., Gosavi, S., Schug, A., Sanbonmatsu, K.Y. and Onuchic, J.N., 2009. An all-atom structure-based potential for proteins: bridging minimal models with all-atom empirical forcefields. Proteins: Structure, Function, and Bioinformatics, 75(2), pp.430-441.

class OpenSMOG.**SBM**(*time_step*, *collision_rate*, *r_cutoff*, *temperature*, *name='OpenSMOG'*)

Bases: object

The *SBM* class performs Molecular dynamics simulations using structure-based models to investigate a broad range of biomolecular dynamics, including domain rearrangements in proteins, folding and ligand binding in RNA, and large-scale rearrangements in ribonucleoprotein assemblies. In its simplest form, a structure-based model defines a particular structure (usually obtained from X-ray, or NMR, methods) as the energetic global minimum.

The *SBM* sets the environment to start the molecular dynamics simulations.

Parameters

- **time_step** (*float, required*) – Simulation time step in units of τ.
- **collision_rate** (*float, required*) – Friction/Damping constant in units of reciprocal time ($1/\tau$).
- **r_cutoff** (*float, required*) – Cutoff distance to consider non-bonded interactions in units of nanometers.
- **temperature** (*float, required*) – Temperature in reduced units.
- **name** (*str*) – Name used in the output files. (Default value: OpenSMOG).

createReporters(*trajectory=True*, *trajectoryName=None*, *energies=True*, *energiesName=None*, *energy_components=False*, *energy_componentsName=None*, *interval=1000*)

Creates the reporters to provide the output data.

Parameters

- **trajectory** (*bool, optional*) – Whether to save the trajectory *.dcd* file containing the position of the atoms as a function of time. (Default value: True).

- **energies** (*bool, optional*) – Whether to save the energies in a *.txt* file containing five columns, comma-delimited. The header of the files shows the information of each collum: #"Step","Potential Energy (kJ/mole)","Kinetic Energy (kJ/mole)","Total Energy (kJ/mole)","Temperature (K)". (Default value: `True`).
- **forces** (*bool, optional*) – Whether to save the potential energy for each applied force in a *.txt* file containing several columns, comma-delimited. The header of the files shows the information of each column. An example of the header is: #"Step","eletrostastic","Non-Contacts","Bonds","Angles","Dihedrals","contact_1-10-12". (Default value: `False`).
- **interval** (*int, required*) – Frequency to write the data to the output files. (Default value: `10**3`)

createSimulation()

Creates the simulation context and loads into the OpenMM platform.

loadGro(*Grofile*)

Loads the *.gro* file format in the OpenMM system platform. The input files are generated using SMOG2 software with the flag `-OpenSMOG`. Details on how to create the files can be found in the SMOG2 User Manual. A tutorial on how to generate the inputs files for the default all-atom and C-alpha models can be found here.

Parameters **Grofile** (*file, required*) – Initial structure for the MD simulations in *.gro* file format generated by SMOG2 software with the flag `-OpenSMOG`. (Default value: `None`).

loadSystem(*Grofile*, *Topfile*, *Xmlfile*)

Loads the input files in the OpenMM system platform. The input files are generated using SMOG2 software with the flag `-OpenSMOG`. Details on how to create the files can be found in the SMOG2 User Manual. A tutorial on how to generate the inputs files for default all-atom and C-alpha models can be found here.

Parameters

- **Grofile** (*file, required*) – Initial structure for the MD simulations in *.gro* file format generated by SMOG2 software with the flag `-OpenSMOG`. (Default value: `None`).
- **Topfile** (*file, required*) – Topology *.top* file format generated by SMOG2 software with the flag `-OpenSMOG`. The topology file lists the interactions between the system atoms except for the "Native Contacts" potential that is provided to OpenSMOG in a *.xml* file. (Default value: `None`).
- **Xmlfile** (*file, required*) – The *.xml* file that contains the all information that defines the "Contact" potential. The *.xml* file is generated by SMOG2 software with the flag `-OpenSMOG`, which support custom potential functions. (Default value: `None`).

loadTop(*Topfile*)

Loads the *.top* file format in the OpenMM system platform. The input files are generated using SMOG2 software with the flag `-OpenSMOG`. Details on how to create the files can be found in the SMOG2 User Manual. A tutorial on how to generate the inputs files for the default all-atom and C-alpha models can be found here.

Parameters **Topfile** (*file, required*) – Topology *.top* file format generated by SMOG2 software with the flag `-OpenSMOG`. The topology file defines the interactions between atoms, except for the "Native Contacts" potential that is provided to OpenSMOG in the form of a *.xml* file. (Default value: `None`).

loadXml(*Xmlfile*)

Loads the *.xml* file format in the OpenMM system platform. The input files are generated using SMOG2 software with the flag `-OpenSMOG`. Details on how to create the files can be found in the SMOG2 User Manual. A tutorial on how to generate the inputs files for default all-atom and C-alpha models can be found here.

Parameters **Xmlfile** (*file, required*) – The *.xml* file that contains all information that defines the "Contact" potentials. The *.xml* file is generated by SMOG2 software with the flag `-OpenSMOG`, which support custom potential energy functions. (Default value: `None`).

printHeader()

run(*nsteps*, *report=True*, *interval=10000*)
Run the molecular dynamics simulation.

Parameters

- **nsteps** (*int, required*) – Number of steps to be performed in the simulation. (Default value: `10**7`)
- **report** (*bool, optional*) – Whether to print the simulation progress. (Default value: `True`).
- **interval** (*int, required*) – Frequency to print the simulation progress. (Default value: `10**4`)

saveFolder(*folder*)
Sets the folder path to save data.

Parameters **folder** (*str, optional*) – Folder path to save the simulation data. If the folder path does not exist, the function will create the directory.

setup_openmm(*platform='opencl'*, *precision='single'*, *GPUindex='default'*, *integrator='langevin'*)
Sets up the parameters of the simulation OpenMM platform.

Parameters

- **platform** (*str, optional*) – Platform to use in the simulations. Opitions are *CUDA*, *OpenCL*, *HIP*, *CPU*, *Reference*. (Default value: `OpenCL`).
- **precision** (*str, optional*) – Numerical precision type of the simulation. Options are *single*, *mixed*, *double*. (Default value: `single`). For details check the OpenMM Documentation.
- **GPUIndex** (*str, optional*) – Set of Platform device index IDs. Ex: 0,1,2 for the system to use the devices 0, 1 and 2. (Use only when GPU != default).
- **integrator** (*str*) – Integrator to use in the simulations. Options are *langevin*, *variableLangevin*, *verlet*, *variableVerlet* and, *brownian*. (Default value: `langevin`).

CHAPTER

FOUR

USING SMOG2 TO GENERATE C-ALPHA AND ALL-ATOM STRUCTURE-BASED MODELS

This tutorial should take between 5 to 10 minutes to complete. Here, we will use the **SMOG2** software pacakge to generate the SBM (Structure-Based Model) input files that will be used to perform a simulation with **OpenSMOG**. To install SMOG2, please check the installation notes in the SMOG 2 user manual, or use the guide here. Details of SMOG2 usage and options are described in the manual. It is assumed that the executable **smog2** is in your path.

4.1 Preparing your PDB file

The following instructions will use a PDB file of CI2 protein (2ci2.pdb).

First, download the PDB file:

```
wget http://www.rcsb.org/pdb/files/2ci2.pdb
```

Then, it is necessary to clean up the file and only keep information needed to define a structure-based model. In this case, let us keep only the ATOM lines:

```
grep "^ATOM" 2ci2.pdb > 2ci2.atoms.pdb
```

Note: Sometimes, you also want HETATMs. This is up to the user. HETATMs can be things that we don't want to include (e.g. HOH), or things that we may want to included (e.g. posttranslational modifications). In this case, we only want ATOM lines.

Next, add an END line to the file 2ci2.atoms.pdb:

```
sed -i -e "\$aEND" 2ci2.atoms.pdb
```

Adjust the file, so that the naming convention conforms with the default SMOG models:

```
smog_adjustPDB -i 2ci2.atoms.pdb -o 2ci2.adj.pdb
```

4.2 Generate OpenSMOG input files for a C-alpha model

Use the adjusted file to generate your input CA model:

```
smog2 -i 2ci2.adj.pdb -CA -dname 2ci2.CA -OpenSMOG
```

4.3 Generate OpenSMOG input files for an all-atom model

To generate input files for the all-atom model, you only need to change the flag -CA to -AA:

```
smog2 -i 2ci2.adj.pdb -AA -dname 2ci2.AA -OpenSMOG
```

Note: When running the simulation in OpenSMOG, there are differences in the simulation protocols and settings. For example, in the case of AA, the cutoff is typically much shorter than the values used with the CA model. However, larger timesteps can typically be used with the AA model. Please, check the C-alpha and All-Atom simulation tutorial pages.

CHAPTER

FIVE

PERFORM A SIMULATION WITH A C-ALPHA STRUCTURE-BASED MODEL USING OPENSMOG

This tutorial should take between 5 to 15 minutes to complete.

Input files for this tutorial can be found here

The first step is to import the **OpenSMOG** module

```
[ ]: from OpenSMOG import SBM
```

SBM class sets the parameters for the simulation:

`name="2ci2"` Sets the name of each simulation *(this name is used as prefix for the outputs)*. `time_step=0.0005` (**reduced time unit**) Sets the time step used in integration. `collision_rate=1.0` (**reduced inverse-time unit**) Sets the collision rate for the Langevin integrator. `r_cutoff=3.0` (**nanometers**) Sets the non-bonded cutoff. `temperature=0.5` (**reduced temperature unit**) Sets the temperature in the simulation.

sbm_CA is an arbitrarily chosen variable name for the SBM object

```
[ ]: sbm_CA = SBM(name='2ci2', time_step=0.0005, collision_rate=1.0, r_cutoff=3.0,␣
     ↪temperature=0.5)
```

There are three hardware platform options to run the simulations:

`platform="cuda"` `platform="HIP"` `platform="opencl"` `platform="cpu"`

if **cuda**, **opencl** or **HIP** is choosen the GPUindex can be defined as "0". If two GPUs are used, one may give "0,1"

```
[ ]: sbm_CA.setup_openmm(platform='cuda',GPUindex='default')
```

Sets the directory name where to save the simulation outputs

```
[ ]: sbm_CA.saveFolder('output_2ci2_CA')
```

Load the **gro**, **top** and **xml** files into the sbm_CA object

```
[ ]: sbm_CA_grofile = 'SMOG2_CA_CI2/2ci2.CA.gro'
     sbm_CA_topfile = 'SMOG2_CA_CI2/2ci2.CA.top'
     sbm_CA_xmlfile = 'SMOG2_CA_CI2/2ci2.CA.xml'

     sbm_CA.loadSystem(Grofile=sbm_CA_grofile, Topfile=sbm_CA_topfile, Xmlfile=sbm_CA_xmlfile)
```

This function returns the name of each contact potential that is being used in the current model. In this example, only a Lennard-Jones-style 10-12 potential is being applied.

The simulation **context** is created with all information given in the previous steps.

```
[ ]: sbm_CA.createSimulation()
```

Create the **reporters** that will save the simulation data in an output folder.

`trajectory=True` Save the trajectory in .dcd format. `energies=True` Save the energy in text format separated by a comma. `interval=10**3` The interval (in steps) at which the trajectory and energies are saved.

```
[ ]: sbm_CA.createReporters(trajectory=True, energies=True, energy_components=True,␣
     ↪interval=10**3)
```

The `run` function receives the following parameters:

`nsteps=10**6` Number of steps to be performed in the simulation. `report=True` Show the simulation details (Progress (%), Step and Time Remaining) `interval=10**3` The step interval to show the details

```
[ ]: sbm_CA.run(nsteps=10**6, report=True, interval=10**3)
```

The output files are located in the output folder

CHAPTER

SIX

PERFORM A SIMULATION WITH AN ALL-ATOM STRUCTURE-BASED MODEL USING OPENSMOG

This tutorial should take between 5 to 15 minutes of reading and performing simulations.

Input files for this tutorial can be found here

The first step is to import the **OpenSMOG** module

```
[ ]: from OpenSMOG import SBM
```

SBM class sets the parameters for the simulation:

`name="2ci2"` Sets the name of each simulation *(this name is used as prefix for the outputs)*. `time_step=0.002` (**reduced time unit**) Sets the time step used in integration. `collision_rate=1.0` (**reduced inverse-time unit**) Sets the collision rate in the Langevin integrator. `r_cutoff=1.2` (**nanometers**) Sets the non-bonded cutoff. `temperature=0.5` (**reduced temperature unit**) Sets the temperature for the simulation.

sbm_AA is an arbitrarily chosen variable name for the SBM object

```
[ ]: sbm_AA = SBM(name='2ci2', time_step=0.002, collision_rate=1.0, r_cutoff=1.2,␣
     ↪temperature=0.5)
```

There are three hardware platform options to run the simulations:

`platform="cuda"` `platform="HIP"` `platform="opencl"` `platform="cpu"`

if **cuda**, **opencl** or **HIP** is choosen the GPUindex can be define as "0". If two GPUs are used, one may give "0,1"

```
[ ]: sbm_AA.setup_openmm(platform='cuda',GPUindex='default')
```

Sets the directory name where the output files are saved

```
[ ]: sbm_AA.saveFolder('output_2ci2_AA')
```

Load the **gro**, **top** and **xml** files into the sbm_AA object

```
[ ]: sbm_AA_grofile = 'SMOG2_AA_CI2/2ci2.AA.gro'
     sbm_AA_topfile = 'SMOG2_AA_CI2/2ci2.AA.top'
     sbm_AA_xmlfile = 'SMOG2_AA_CI2/2ci2.AA.xml'

     sbm_AA.loadSystem(Grofile=sbm_AA_grofile, Topfile=sbm_AA_topfile, Xmlfile=sbm_AA_xmlfile)
```

This function returns the name of each contact potential that is being used in the current model. In this example, only a Lennard-Jones-style 6-12 potential is being applied.

The simulation **context** is created with all information given in the previous steps.

[ ]:
```
sbm_AA.createSimulation()
```

Create the **reporters** that will save the simulation data in an output folder.

`trajectory=True` Save the trajectory in .dcd format. `energies=True` Save the energy in text format separated by a comma. `interval=10**3` The interval (in steps) at which the trajectory and energies are saved.

[ ]:
```
sbm_AA.createReporters(trajectory=True, energies=True, energy_components=True,␣
↪interval=10**3)
```

The `run` function receives the following parameters:

`nsteps=10**6` Number of steps to perform in the simulation. `report=True` Shows the simulation details (Progress (%), Step and Time Remaining) `interval=10**3` The step interval to show the details

[ ]:
```
sbm_AA.run(nsteps=10**6, report=True, interval=10**3)
```

The output files are located in the output folder

CHAPTER

SEVEN

HOW TO CITE OPENSMOG

When using the SMOG2-OpenSMOG framework for publications, please use the following citations:

```
@article{SMOG2,
title = {SMOG 2: A Versatile Software Package for Generating Structure-Based Models},
author = {Noel, Jeffrey K. and
Levi, Mariana and
Raghunathan, Mohit and
Lammert, Heiko and
Hayes, Ryan L. and
Onuchic, Jos{\'{e}} N. and
Whitford, Paul C.},
journal = {PloS Comp Biol},
pages = {e1004794},
volume = {12},
year = {2016}
}

@article{OpenSMOG,
title={SMOG 2 and openSMOG: Extending the limits of structure-based models},
author={Oliveira Jr, Antonio B  and Contessoto, Vinicius G and
Hassan, Asem and
Byju, Sandra and
Wang, Ailun and
Wang, Yang and
Dorero-Rojas, Esteban and
Mohanty, Udayan and
Noel, Jeffrey K and
Onuchic, Jose N and
Whitford, Paul C},
journal={bioRxiv},
doi={10.1101/2021.08.15.456423}
}
```

CHAPTER

EIGHT

REFERENCES

- Jeffrey K Noel, Mariana Levi, Mohit Raghunathan, Heiko Lammert, Ryan L Hayes, José N Onuchic, and Paul C Whitford. Smog 2: a versatile software package for generating structure-based models. *PLoS computational biology*, 12(3):e1004794, 2016.
- Cecilia Clementi, Hugh Nymeyer, and José Nelson Onuchic. Topological and energetic factors: what determines the structural details of the transition state ensemble and "en-route" intermediates for protein folding? an investigation for small globular proteins. *Journal of molecular biology*, 298(5):937–953, 2000.
- Paul C Whitford, Jeffrey K Noel, Shachi Gosavi, Alexander Schug, Kevin Y Sanbonmatsu, and José N Onuchic. An all-atom structure-based potential for proteins: bridging minimal models with all-atom empirical forcefields. *Proteins: Structure, Function, and Bioinformatics*, 75(2):430–441, 2009.
- Jeffrey K Noel, Paul C Whitford, and José N Onuchic. The shadow map: a general contact definition for capturing the dynamics of biomolecular folding and function. *The Journal of Physical Chemistry B*, 116(29):8692–8702, 2012.
- Heiko Lammert, Alexander Schug, and José N Onuchic. Robustness and generalization of structure-based models for protein folding and function. *Proteins: Structure, Function, and Bioinformatics*, 77(4):881–891, 2009.
- Jeffrey K Noel and José N Onuchic. The many faces of structure-based potentials: from protein folding landscapes to structural characterization of complex biomolecules. In *Computational modeling of biological systems*, pages 31–54. Springer, 2012.
- Mariana Levi, Prasad Bandarkar, Huan Yang, Ailun Wang, Udayan Mohanty, Jeffrey K Noel, and Paul C Whitford. Using smog 2 to simulate complex biomolecular assemblies. In *Biomolecular Simulations*, pages 129–151. Springer, 2019.
- Antonio B Oliveira Jr, Vinicius G Contessoto, Matheus F Mello, and Jose N Onuchic. A scalable computational approach for simulating complexes of multiple chromosomes. *Journal of Molecular Biology*, 433(6):166700, 2020.

CHAPTER

NINE

LICENSE

```
MIT License

Copyright (c) 2020-2021 The Center for Theoretical Biological Physics (CTBP) - Rice␣
↪University & Northeastern University

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

CHAPTER

TEN

INDICES AND TABLES

- genindex
- modindex
- search

BIBLIOGRAPHY

[1] Jeffrey K Noel, Mariana Levi, Mohit Raghunathan, Heiko Lammert, Ryan L Hayes, José N Onuchic, and Paul C Whitford. Smog 2: a versatile software package for generating structure-based models. *PLoS computational biology*, 12(3):e1004794, 2016.

[2] Cecilia Clementi, Hugh Nymeyer, and José Nelson Onuchic. Topological and energetic factors: what determines the structural details of the transition state ensemble and "en-route" intermediates for protein folding? an investigation for small globular proteins. *Journal of molecular biology*, 298(5):937–953, 2000.

[3] Paul C Whitford, Jeffrey K Noel, Shachi Gosavi, Alexander Schug, Kevin Y Sanbonmatsu, and José N Onuchic. An all-atom structure-based potential for proteins: bridging minimal models with all-atom empirical forcefields. *Proteins: Structure, Function, and Bioinformatics*, 75(2):430–441, 2009.

PYTHON MODULE INDEX

INDEX