##Contents

1. What you will learn 3
2. Table of Contents 5
3. Indices and tables 13
Molecular dynamics tutorial presenting possibilities given by vanilla python and open source Python libraries.

Tutorial covers in details creating and analyzing trajectories produced by the most popular simulation packages. It also contains several good practices borrowed from software engineering like: version control, workflow automation, data pipelines, and scaling up / scaling out computation.
CHAPTER 1

What you will learn

• what are most popular tools to do MD
• popular data formats and python tools to work with them
• how to prepare robust and easy to set up work environment
• how to build MD system
• how to run MD simulations using most popular packages
• how to dock small ligands
• how to scale up/out your simulations
• how to load a structure or a MD trajectory (popular formats)
• how to select parts of your system
• how to work with atoms, residues and molecules
• how to deal with large data-sets (bigger than RAM)
• how to analyze MD trajectories
• how to write out or convert trajectories
• how to use various algorithms in analysis (clusterization, structure, ML)
• how to build interactive visualizations
Python MD Swiss Army Tool-Kit

Most of the modern tools for bio-engineering are gathered in OMNIA_ repository maintained by Continuum Analytics.

OpenMM & OpenMMtools

http://openmm.org/documentation.html

A toolkit for molecular simulation using high performance GPU code. OpenMM consists of two parts:

A set of libraries that lets programmers easily add molecular simulation features to their programs An “application layer” that exposes those features to end users who just want to run simulations

MDTraj

http://mdtraj.org/1.8.0/

A modern, open library for the analysis of molecular dynamics trajectories.

Read, write and analyze MD trajectories with only a few lines of Python code.

MDTraj is a python library that allows users to manipulate molecular dynamics (MD) trajectories. Features include:

Wide MD format support, including pdb, xtc, trr, dcd, binpos, netcdf, mdcdf, prmtop, and more. Extremely fast RMSD calculations (4x the speed of the original Theobald QCP). Extensive analysis functions including those that compute bonds, angles, dihedrals, hydrogen bonds, secondary structure, and NMR observables. Lightweight, Pythonic API. MDTraj includes a command-line application, mdconvert, for converting trajectories between formats.
MDAnalysis

MDAnalysis is an object-oriented Python library to analyze trajectories from molecular dynamics (MD) simulations in many popular formats. It can write most of these formats, too, together with atom selections suitable for visualization or native analysis tools.

*MDAnalysis* allows one to read particle-based trajectories (including individual coordinate frames such as biomolecules in the PDB format) and access the atomic coordinates through NumPy arrays. This provides a flexible and relatively fast framework for complex analysis tasks. In addition, powerful atom selection commands are implemented. Trajectories can also be manipulated (for instance, fit to a reference structure) and written out.

**yank**

Linux only.

http://getyank.org/0.15.2.dev0/#

https://anaconda.org/omnia/yank-examples

**Features:**

- Modular Python framework to facilitate development and testing of new algorithms
- GPU-accelerated via the OpenMM toolkit (see benchmarks)
- Alchemical free energy calculations in both explicit and implicit solvent
- Hamiltonian exchange among alchemical intermediates with Gibbs sampling framework
- General Markov chain Monte Carlo framework for exploring enhanced sampling methods
- Built-in equilibration detection and convergence diagnostics
- Support for AMBER prmtop/inpcrd files
- Support for absolute binding free energy calculations
- Support for transfer free energies (such as hydration or partition free energies)
simpletraj

https://github.com/arose/simpletraj

Lightweight coordinate-only trajectory reader based on code from GROMACS, MDAnalysis, VMD. Linux only.

parmed


ParmEd is a general tool for aiding in investigations of biomolecular systems using popular molecular simulation packages, like Amber, CHARMM, and OpenMM written in Python.

There are two parts to ParmEd – the Python API that exposes the core classes used in its modeling capabilities, and two front-end Python programs (parmed and its GUI counterpart, xparmed) that make use of the ParmEd API to allow rapid prototyping and parameter-topology modifications for use in molecular simulations.

pdbfixer


PDBFixer fixes problems in PDB files to prepare them for molecular simulation (development snapshot).

Protein Data Bank (PDB or PDBx/mmCIF) files often have a number of problems that must be fixed before they can be used in a molecular dynamics simulation. The details vary depending on how the file was generated. Here are some of the most common ones: If the structure was generated by X-ray crystallography, most or all of the hydrogen atoms will usually be missing. There may also be missing heavy atoms in flexible regions that could not be clearly resolved from the electron density. This may include anything from a few atoms at the end of a sidechain to entire loops. Many PDB files are also missing terminal atoms that should be present at the ends of chains. The file may include nonstandard residues that were added for crystallography purposes, but are not present in the naturally occurring molecule you want to simulate. The file may include more than what you want to simulate. For example, there may be salts, ligands, or other molecules that were added for experimental purposes. Or the crystallographic unit cell may contain multiple copies of a protein, but you only want to simulate a single copy. There may be multiple locations listed for some atoms. If you want to simulate the structure in explicit solvent, you will need to add a water box surrounding it. PDBFixer can fix all of these problems for you in a fully automated way. You simply select a file, tell it which problems to fix, and it does everything else. PDBFixer can be used in three different ways: as a desktop application with a graphical user interface; as a command line application; or as a Python API. This allows you to use it in whatever way best matches your own needs for flexibility, ease of use, and scriptability. The following sections describe how to use it in each of these ways.

bhmm

https://pypi.python.org/pypi/bhmm/0.5.1

A toolkit for Bayesian hidden Markov model analysis of single-molecule trajectories

This project provides tools for estimating the number of metastable states, rate constants between the states, equilibrium populations, distributions characterizing the states, and distributions of these quantities from single-molecule data. This data could be FRET data, single-molecule pulling data, or any data where one or more observables are recorded as a function of time. A Hiddenstclu Markov Model (HMM) is used to interpret the observed dynamics, and a distribution of models that fit the data is sampled using Bayesian inference techniques and Markov chain Monte Carlo (MCMC), allowing for both the characterization of uncertainties in the model and modeling of the expected information gain by new experiments.
fastcluster

https://pypi.python.org/pypi/fastcluster

Fast hierarchical clustering routines for R and Python.

This library provides Python functions for hierarchical clustering. It generates hierarchical clusters from distance matrices or from vector data.

conda install -c omnia fastcluster

pint

https://pint.readthedocs.io/en/0.7.2/

Pint is a Python package to define, operate and manipulate physical quantities: the product of a numerical value and a unit of measurement. It allows arithmetic operations between them and conversions from and to different units.

It is distributed with a comprehensive list of physical units, prefixes and constants. Due to its modular design, you can extend (or even rewrite!) the complete list without changing the source code. It supports a lot of numpy mathematical operations without monkey patching or wrapping numpy. Physical quantities module.

cpptraj

Linux only.

pacmol


One of the biggest issues you face when you first start doing molecular dynamics (MD) simulations is how to create an initial geometry that won’t blow up in the first few time steps. Repulsive forces are very steep if the atoms are too close to each other, and if you are trying to simulate a condensed phase (liquid, solid, or interfacial) system, it can be hard to know how to make a sensible initial structure.

deepchem

https://github.com/deepchem/deepchem

Deep-learning models for drug discovery and quantum chemistry.

DeepChem is a python library that aims to make the use of machine-learning in drug discovery straightforward and convenient. http://deepchem.io

linux only

pyvisfile

File formats

HDF5
netCDF
xmc

Hydrogen Bonds Analysis

Identify hydrogen bonds based on cutoffs for the Donor-H...Acceptor distance and angle. The criterion employed is

\[ \theta > 120\] and \[ r_{\text{extH...Acceptor}} < 2.5\text{Å} \]. When donor the donor is ‘N’ and the acceptor is ‘O’, this corresponds to the definition established in [1]. The donors considered by this method are NH and OH, and the acceptors considered are O and N. Parameters

- traj : md.Trajectory
- freq [float, default=0.1] Return only hydrogen bonds that occur in greater this fraction of the frames in the trajectory.
- exclude_water [bool, default=True] Exclude solvent molecules from consideration
- periodic [bool, default=True] Set to True to calculate displacements and angles across periodic box boundaries.
- sidechain_only [bool, default=False] Set to True to only consider sidechain-sidechain interactions.
- hbonds [np.array, shape=[n_hbonds, 3], dtype=int] An array containing the indices atoms involved in each of the identified hydrogen bonds. Each row contains three integer indices, \((d_i, h_i, a_i)\), such that \(d_i\) is the index of the donor atom, \(h_i\) the index of the hydrogen atom, and \(a_i\) the index of the acceptor atom involved in a hydrogen bond which occurs (according to the definition above) in proportion greater than \(freq\) of the trajectory.

Each hydrogen bond is distinguished for the purpose of this function by the indices of the donor, hydrogen, and acceptor atoms. This means that, for example, when an ARG sidechain makes a hydrogen bond with its NH2 group, you might see what appear like double counting of the h-bonds, since the hydrogen bond formed via the H_1 and H_2 are counted separately, despite their “chemical indistinguishably” Examples

```
md.baker_hubbard(t) array([[ 0, 10, 8],
                            [ 0, 11, 7], [ 69, 73, 54], [ 76, 82, 65], [119, 131, 89], [140, 148, 265], [166, 177, 122], [181, 188, 231]])
```

```
>>> label = lambda hbond : '%s -- %s' % (t.topology.atom(hbond[0]), t.topology.atom(hbond[2]))
>>> for hbond in hbonds:
...     print label(hbond)
GLU1-N -- GLU1-OE2
GLU1-N -- GLU1-OE1
GLY6-N -- SER4-O
CYS7-N -- GLY5-O
TYR11-N -- VAL8-O
MET12-N -- LYS20-O
```

B-factor field

http://www.mdanalysis.org/MDAnalysisTutorial/writing.html

Native Contacts

http://mdtraj.org/1.8.0/examples/native-contact.html

Principal components analysis (PCA)

...ramachandran:

Ramachandran Plot

RMSD

RMSF

Solvent accessibility

We’ll use the algorithm from Shrake and Rupley for computing the SASA.

http://mdtraj.org/1.8.0/examples/solvent-accessible-surface-area.html

Notes

This code implements the Shrake and Rupley algorithm, with the Golden Section Spiral algorithm to generate the sphere points. The basic idea is to get a mesh of points representing the surface of each atom (at a distance of the van der waals radius plus the probe radius from the nuclei), and then count the number of such mesh points that are on the molecular surface – i.e. not within the radius of another atom. Assuming that the points are evenly distributed, the number of points is directly proportional to the accessible surface area (its just 4*pi*r^2 time the fraction of the points that are accessible).

There are a number of different ways to generate the points on the sphere – possibly the best way would be to do a little “molecular dyanmics” : put the points on the sphere, and then run MD where all the points repel one another and wait for them to get to an energy minimum. But that sounds expensive.

This code uses the golden section spiral algorithm (picture at http://xisupport.com/2012/02/25/evenly-distributing-points-on-a-sphere-with-the-golden-sectionspiral/) where you make this spiral that traces out the unit sphere and then put points down equidistant along the spiral. It’s cheap, but not perfect.

The gromacs utility g_sas uses a slightly different algorithm for generating points on the sphere, which is based on an icosahedral tesselation. roughly, the icosahedral tesselation works something like this http://www.ziyan.info/2008/11/sphere-tessellation-using-icosahedron.html

Waals value is known, a default of 2 angstroms is used. However, because certain atoms in biophysical simulations have a high chance of being completely ionized, we have decided to give the following atoms their ionic radii:

- **+2**: Be, Mg, Ca, Ba
- **+1**: Li, Na, K, Cs
- **-1**: Cl

These ionic radii were taken from: Shannon, R. D. Revised effective ionic radii and systematic studies of interatomic distances in halides and chalcogenides. Acta Crystallographica Section A 32, 751–767 (1976). doi:10.1107/S0567739476001551

For most atoms, adding electrons usually doesn’t change the radius much (<10%), while removing them changes it substantially (>50%). Further, when atoms like N, S, and P, are positive, they are bound to atoms in such a way that would “hide” their radii anyway. We have therefore chosen to just use their vdW radii.

**Protein visualizations**

**UCSF Chimera**

[https://www.cgl.ucsf.edu/chimera/](https://www.cgl.ucsf.edu/chimera/)

UCSF Chimera is a highly extensible program for interactive visualization and analysis of molecular structures and related data, including density maps, supramolecular assemblies, sequence alignments, docking results, trajectories, and conformational ensembles. High-quality images and animations can be generated. Chimera includes complete documentation and several tutorials, and can be downloaded free of charge for academic, government, nonprofit, and personal use.

**nglview**

IPython widget to interactively view molecular structures and trajectories.

**Data visualization**

**Bokeh**

**bqplot**

**Docking substrate**

**OpenMM simulation**

[http://mdtraj.org/1.8.0/examples/openmm.html](http://mdtraj.org/1.8.0/examples/openmm.html)
CHAPTER 3

Indices and tables

- genindex
- modindex
- search