Molecular dynamics analysis libraries, p. 2

with an example based on the dynamics in the physiopathology of gelsolin

Toni Giorgino
toni.giorgino@cnr.it
www.giorginolab.it
@giorginolab

Part I. Motivation

Finnish-type amyloidogenic gelsolin variant -
an example of protein dynamics playing a role in
proteotoxicity and drug design discovered by MD.

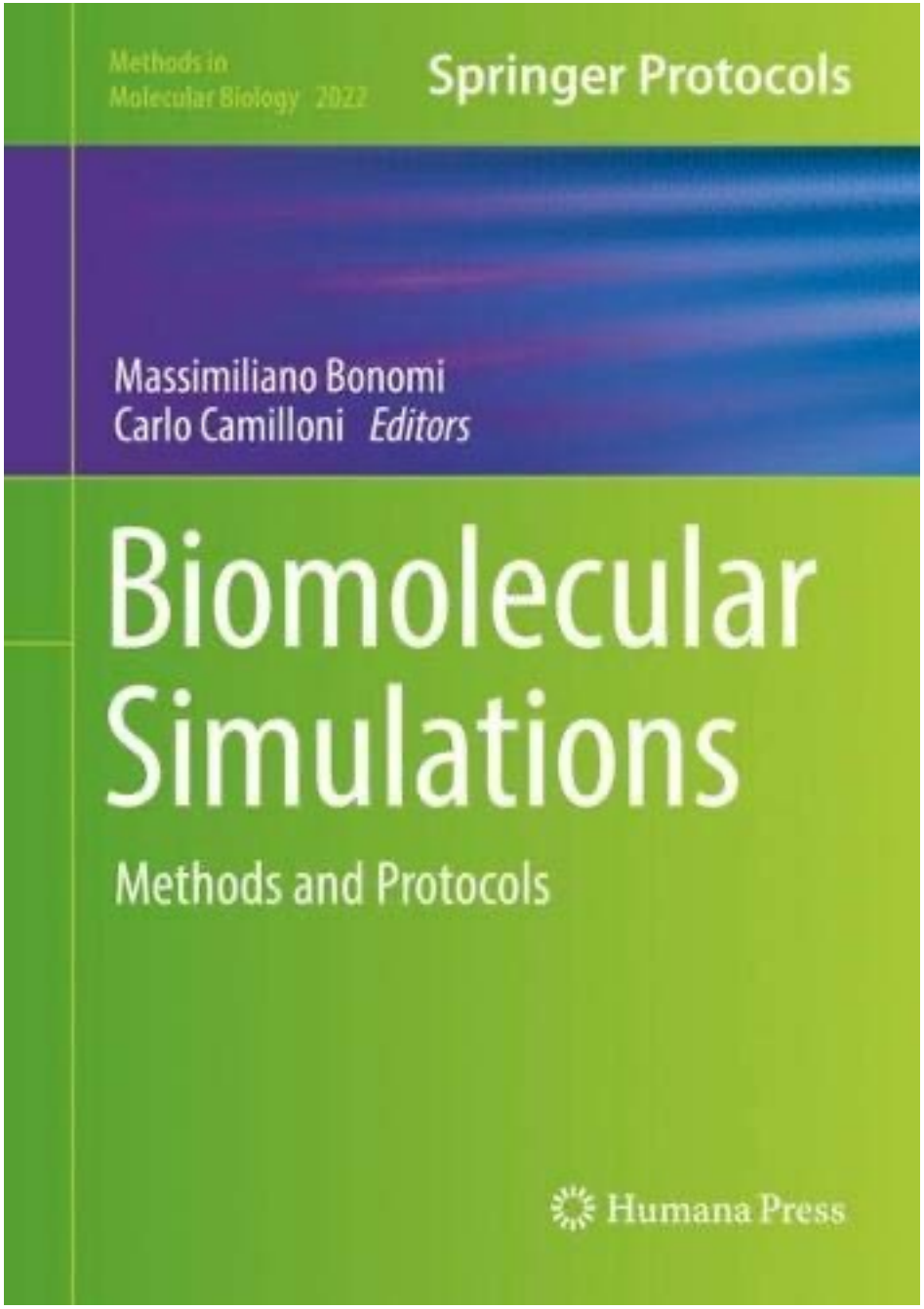
Part II. Practice

MD analysis libraries: intro and reproduction of
the analysis* shown in the paper.

* Marked with
this symbol →

Part II.
MD analysis libraries

(Practice)



Springer, ISBN 978-1-4939-9608-7
Preprint here.

Check for updates

Chapter 20

Analysis Libraries for Molecular Trajectories: A Cross-Language Synopsis

Toni Giorgino

Abstract

Analyzing the results of molecular dynamics (MD)-based simulations usually entails extensive manipulations of file formats encoding both the topology (e.g., the chemical connectivity) and configurations (the trajectory) of the simulated system. This chapter reviews a number of software libraries developed to facilitate interactive and batch analysis of MD results with scripts written in high-level, interpreted languages. It provides a beginners' introduction to MD analysis presenting a side-by-side comparison of major scripting languages used in MD and shows how to perform common analysis tasks within the Visual Molecular Dynamics (VMD), Bio3D, MDTraj, MDAnalysis, and High-Throughput Molecular Dynamics (HTMD) environments.

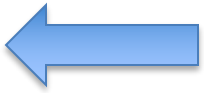
Key words Molecular dynamics, Trajectory analysis, Scripting languages, VMD, Bio3D, MDTraj, MDAnalysis, HTMD

1 Introduction

The backbone of molecular dynamics (MD)-based methods is to integrate the equations of motion of a system with a given Hamiltonian. The integration is performed by an MD engine with a finite time-step, sufficiently fine to capture the fastest motion of interest (e.g., bond vibrations). Commonly, one is interested in long-time behavior, and therefore, simulations are performed for several orders of magnitudes longer than the integration time-steps, making integration the most compute-intensive component of the MD workflow; this, in turn, makes it natural to keep a record ("trajectory") of the states through which the system goes for later analysis.

The objective of this chapter is to provide an operative introduction to the libraries most often used in MD analysis in combination with the corresponding programming languages. In particular, I strive to provide (a) a side-by-side view of the constructs most important for analysis (including file input and output

Analysis of MD trajectories

- Interactive: VMD, Chimera, PyMol…
  - Intuitive
  - Suitable for one-off tasks
- Scripted: for…
  - repeated analysis (e.g. ensembles)
  - custom tasks (your own ideas)
  - automated analysis, e.g. machine learning
- Analysis libraries are needed 

MD analysis libraries

Library	Language
VMD	TCL
Bio3D	R
MDAnalysis	Python
MDTraj	Python
HTMD/MoleculeKit	Python

- We'll show examples for **MDTraj**,
  but there are *direct* equivalents in the others.
- In fact, converting is a useful exercise.
- See the Chapter.

1. **Please clone or download…**

which contains the papers and **data files**.

2. **Open the Colaboratory link** 

It is a "live" Python notebook to run your code on Google's servers. (Account needed)

Alternatively, work locally on your PC.

Loading a trajectory

- First, import (activate) the library
- Then, load the *topology* and *trajectory**

VMD

```
set t [mol new $pdb]
animate delete all
mol addfile $xtc waitfor all
```

MDAnalysis

```
import MDAnalysis as mda
t = mda.Universe(pdb, xtc)
```

HTMD

```
from htmd.ui import *
t=Molecule(pdb)
t.read(xtc)
```

Bio3D

```
library(bio3d)
tp <- read.pdb(pdb)
tp$xyz <- read.dcd(dcd)
```

MDTraj

```
import mdtraj as mdt
t = mdt.load(xtc, top=pdb)
```

* Atom names, types, bonds, etc.
Usually a PDB or PSF file.

Several formats are supported.
Here we use PDB+XTC.

VMD

```
# Number of frames
molinfo top get numframes

set t [atomselect top all]
$t num;          # Number of atoms

$t frame 0
$t get {x y z}; # Coordinates

pbc get;         # Unit cell
```

MDAnalysis

```
# Self-explanatory
t.atoms.n_atoms
t.trajectory.n_frames

# Atoms by 3
t.atoms.positions

# Unit cell
t.atoms.dimensions
```

HTMD

```
t.numFrames
t.numAtoms

# Atoms by 3 by frames
t.coords

# Unit cell
t.box[:,0]
```

Bio3D

```
nrow(tp$xyz)        # 40 frames
nrow(tp$atom)       # 28799 atoms

## Accessing coordinates in frame 0
## reshaped for convenience
xyz <- tp$xyz[1,]
xyz <- matrix(xyz, ncol=3, byrow=T)

## Or: array(xyz,c(40,3,28799))
```

MDTraj

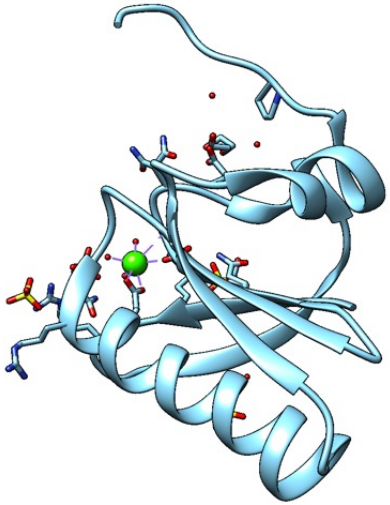
```
# Number of frames
len(t)

# Frames by Atoms by 3
t.xyz.shape
# Coordinates in frame 0
t.xyz[0]

# Unit cell
t.unitcell_lengths[0,:]
```

Access molecular data

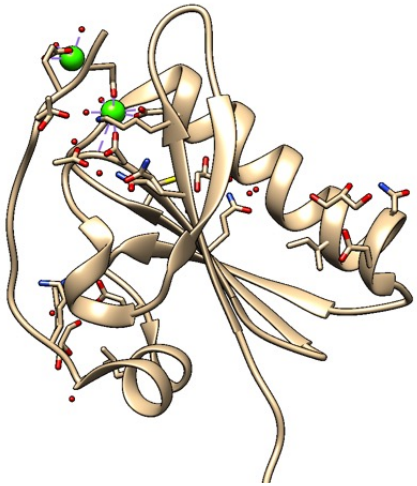
Alignment

Calculations are often performed *after* a rigid transformation which *optimally* superimposes two structures (or two frames). It removes diffusion.

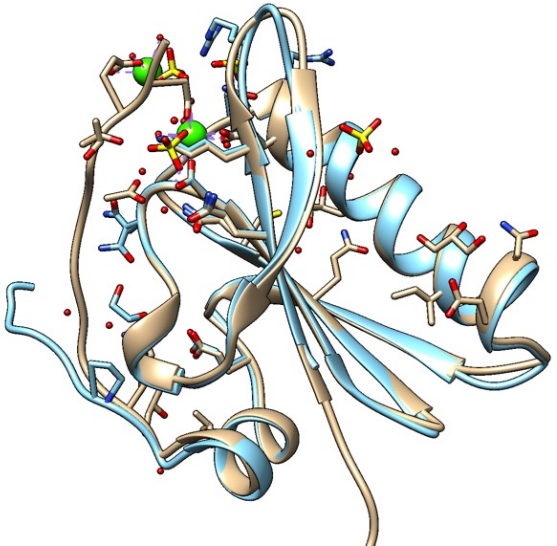
Before



Minimize RMSD



After



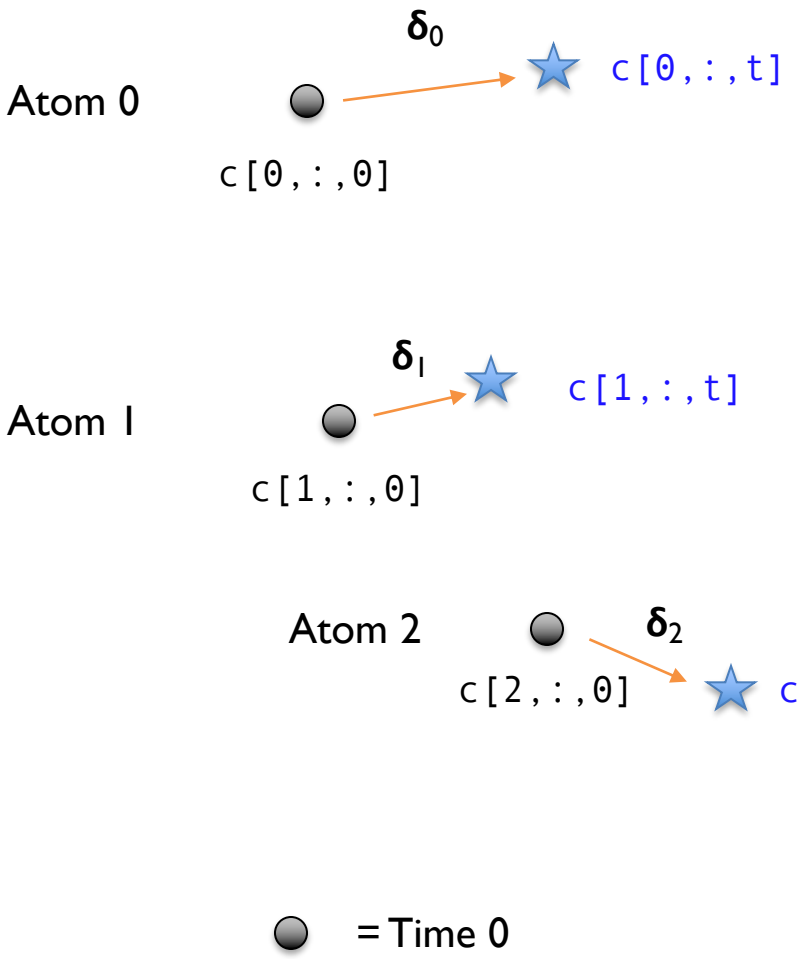
RMSD

Is the *mean squared* displacement between two sets of atoms

$$\mathrm{RMSD}(\mathbf{x}, \mathbf{y}) = \sqrt{\frac{\sum_{i=1}^{N_{\mathrm{atoms}}} (\mathbf{x}_i - \mathbf{y}_i)^2}{N_{\mathrm{atoms}}}}$$

RMSD

Coordinates array: `coords[ atom_index , axis , frame ]`



$\boldsymbol{\delta}_i$ = `c[:,:,t]-c[:,:,0]`
(vectors)

$|\boldsymbol{\delta}_i| = \sqrt{\delta_{ix}^2 + \delta_{iy}^2 + \delta_{iz}^2}$

Data files (directory `data`)

	Apo form	In complex with nanobody Nb11
Wild type	`WT`	`WT+Nb`
D187N mutant	`D187N`	`D187N+Nb`

For each combination you will find:

- a PDB file
- a PSF file
- an *unwrapped* trajectory in XTC format (10 ns/frame)

IF present, Nb was held restrained

Now we open the Colaboratory Notebook and try to solve the exercises. ☺

Advanced: rewrite it to use a different library.