ProDy

Protein Dynamics & Sequence Analysis

# coMD Tutorial

***Release***

**Cihan Kaya**

December 04, 2024

Learn global transition pathways with multiscale hybrid methodology, collective molecular dynamics.

CHAPTER

ONE

# INTRODUCTION

Collective Molecular Dynamics is a **VMD_** plugin GUI and a Python module developed for setup and analysis of simulations described in *[MG13]*.

## 1.1 Installation

1. **VMD_** 1.9.2 or later is required for using GUI. NAMD[1] is required for running molecular dynamics simulations and latest version of **ProDy_** is required for elastic network model calculations. Following are required for performing coMD calculations:

- **Python_** 2.7
- **NumPy_** 1.3 or later

2. Download one of the following archive files:

- comd_plugin_files.tgz
- comd_plugin_files.zip

3. Extract contents of the archive and copy `comd` folder to VMD TCL plugins directory, i.e. `$VMDDIR/plugins/noarch/tcl/`.

   Then, insert following line into `$VMDDIR/scripts/vmd/loadplugins.tcl` at line 200:

```
vmd_install_extension comd comd_tk "Simulations/Collective Molecular Dynamics (CoMD)"
```

   If you are not sure where VMD directory is located, run **vmd**, and type the following command line in the VMD console:

```
global env; puts $env(VMDDIR)
```

## 1.2 coMD Plugin

Collective Molecular Dynamics (coMD) plugin, shown below, has a panel to setup, collective molecular dynamics simulations:


[1] http://www.ks.uiuc.edu/Research/namd/

The rest of the tutorial will show you how to use this panel, and described required inputs and outputs from analysis steps.

## 1.3 Tutorial Files

Files in the following archives can be used to follow this tutorial:

- coMD Tutorial Files (TGZ)
- coMD Tutorial Files (ZIP)

Here is a list of these files:

```
350K Feb 29  2024 1ake.pdb
304K Feb 29  2024 4ake.pdb
2.8M Feb 29  2024 final_ionized.pdb
 17M Feb 29  2024 final_trajectory.dcd
2.2M Feb 29  2024 initial_ionized.pdb
 14M Feb 29  2024 initial_trajectory.dcd
```

## 1.4 How to Cite

If you benefited from collective molecular dynamics in your research, please cite the following paper:

CHAPTER

TWO

# BACKGROUND

Biological functions of biomolecules is accomplished by conformational motions that enable the molecules to sample functional substates. Some of these substates are short-lived, and therefore difficult (or even impossible) to detect experimentally. Here, computational tools are useful to predict plausible transition pathways, which can be probed experimentally. They may also provide insights into accessible conformational states, or the energy landscape, which often determines the structural changes taking place during interactions with other proteins and ligands. Full atomic MD simulations often fail in sampling large-scale cooperative changes in conformation (global motions) especially in the case of large molecular systems. Coarse-grained approaches like ANM fail in providing atomic details, while they provide an accurate description of global dynamics. We have developed a hybrid method, coMD, that takes advantage of the ability of ANM to sample global modes so as to direct MD simulations. This way we can sample global motions at full atomic resolution. Each step in coMD is a collective movement of the entire molecule along a given ANM mode, selected by a Monte Carlo/Metropolis algorithm.

CHAPTER

THREE

# METHODOLOGY

Below is a schematic description of the methodology for sampling the transition pathway between two known structures A and B. The same approach can be used for sampling the conformational space in the vicinity of both substates, simply by defining the Metropolis criteria to accept all moves, irrespective of the energy change at the last step. Thus, the method can be used for either characterizing the pathway(s) between two endpoints A and B, or mapping the conformational space in general for exploring alternative substates.


In a nutshell, the method consists of the following steps applied to each endpoint in parallel: (i) selecting a given ANM mode using a Monte Carlo algorithm based on the weight (inverse eigenvalue, square root) of each mode, (ii) examining if this mode allows for approaching the other endpoint, and accepting/rejecting it with the help of a Metropolis algorithm based on an energy function that penalizes the increase in RMSD from the endpoint (this condition can be removed to allow for unconstrained sampling), (iii) generate a target conformation based on the selected mode, (iv) use this target conformation to simulate (by TMD) the transition toward this target conformation - which defines an intermediate conformer in the process of sampling the transition path (or the conformational space), (v) energy minimize, and (vi) go back to step (i) by redefining the intermediate conformation as the new endpoint, until completion of the sampling (e.g. if the two endpoints generated in parallel are sufficiently close to each other).

CHAPTER

FOUR

# SIMULATION SETUP

A system that contains protein structures with water and counter ions in collective molecular dynamics simulations can be prepared using the following interface:


## 4.1 Input Files

`.pdb` files for both initial and final structure of the protein are required from the user. You can learn how to prepare these files from NAMD tutorials[3].

## 4.2 Protein Structures

1. First, select initial `.pdb` (structure) files and final `.pdb` (structure) files. You can use the adelynate kinase files provided for this tutorial (1ake.pdb, 4ake.pdb). Alternatively, structure and coordinate files for a protein of interest can also be used. These files should contain all atoms required for protein stability and function, these may include cofactors and metal atoms. Crystallographic water molecules may also be retained in the structure.
2. Select the corresponding chain IDs for initial and final structure. For adenylate kinase both chain IDs should be 'A'. You can also select chains containing non-protein components such as membrane. If you leave this box blank, all protein atoms will be selected.

## 4.3 Ionization Parameters

1. To build the topology of the structure, a topology file needs to be provided. This file will parametrize the topology of structures e.g. bond lengths and bond angles. If not given, it will use CHARMM36 all-hydrogen topology file for proteins and lipids. For more information visit Topology[4] pages.
2. Enter simulation box padding (distance from the protein to box surfaces) in x, y and z and select whether you would like to add counter ions and what concentration. Collective Molecular Dynamics GUI uses Solvate[5] and Autoionize[6] plugins to add water and ions.

[3] http://www.ks.uiuc.edu/Training/Tutorials/
[4] http://www.ks.uiuc.edu/Training/Tutorials/namd/namd-tutorial-unix-html/node24.html
[5] http://www.ks.uiuc.edu/Research/vmd/plugins/solvate/
[6] http://www.ks.uiuc.edu/Research/vmd/plugins/autoionize/

## 4.4 Minimization Parameters

1. For every cycle, the structures (protein and water) will be minimized by using NAMD. For parametrization of molecular dynamics simulation in NAMD, an input file is required. If not given, it will use CHARMM36 all-hydrogen parameter files. For more information visit Parameter[7] pages.
2. For minimization, two additional parameters are required. The first one is the temperature at which the minimization will be performed and the second one is the total number of steps to perform minimization.

## 4.5 Conformational Sampling with ANM Modes

For sampling of conformations with ANM modes, there are some optional parameters that all take default (recommended) values.

ANM cutoff is the parameter to determine the contacts on the proteins for ANM calculations (default 15).

The second parameter is the maximum deviation for each step - this would be the RMSD for the first (largest amplitude) mode. From this, we derive a scale factor for perturbing structure in the direction of a given ANM mode. The default is 2 A.

Other important parameters are the acceptance ratio (default 0.9), which determines how many wrong direction moves are accepted versus right direction ones, and the maximum number of steps (default 100).

## 4.6 Targeted Molecular Dynamics Parameters

For every cycle, after conformational sampling with ANM modes towards to the other structure, the side chains of the protein need to move by performing a fast targeted molecular dynamics (TMD) simulation. In TMD simulations, two parameters are required. The first one is the spring constant (kcal/mol/A^2) and the second one is the length of TMD simulation (ps). For further information visit TMD[8] pages.

## 4.7 Simulation Options

coMD simulations will run in cycles and the number of cycles is the global parameter for them. The simulation will run until the number of cycles is achieved or the RMSD between initial and final structure is less than 1.5 A or the reduction in the RMSD is less than 0.15 A. Also provided in this block are which GPUs and the number of physical cores to use for parallel NAMD simulations, and the option to run now or not. You would not run now if you intended to copy the files to a cluster for example.

## 4.8 Output Options

Here you select an output folder and a file prefix to distinguish different runs. Performing multiple simulations to see whether results are reproducible is always a good idea.

## 4.9 Output Files

There will be an output folder with a python script (.py file), a tcl script (.tcl file), and two .pdb and .psf files. The tcl script should run from vmd.

For a summary of contents of the final system, see `prefix.log` file.

[7] http://www.ks.uiuc.edu/Training/Tutorials/namd/namd-tutorial-unix-html/node25.html
[8] http://www.ks.uiuc.edu/Research/namd/2.10b1/ug/node47.html

## 4.10 Simulation

Now you need to run coMD simulations. To perform those simulations it is possible to use the following command:

```
vmd -dispdev text -e your_output_prefix.tcl
```

If you want to run this simulation on a cluster, copy this directory on cluster and put the command on the queue system.

When simulations are complete, you can continue with following analysis steps.

CHAPTER

FIVE

# COLLECTIVE MOLECULAR DYNAMICS ANALYSIS

After running a coMD simulation, the results will be prepared in the same folder named in the setup section. You need to download the following files from that folder:

1. `initial_ionized.pdb`
2. `final_ionized.pdb`
3. `initial_trajectory.dcd`
4. `final_trajectory.dcd`

## 5.1 Preparing Trajectory Files

We recommend that you will follow this tutorial by typing commands in an IPython session, e.g.:

```
$ ipython
```

or with pylab environment:

```
$ ipython --pylab
```

First, we will make necessary imports from ProDy and Matplotlib packages.

```
In [1]: from prody import *

In [2]: from pylab import *

In [3]: ion()
```

coMD simulations will create two different trajectories and we need to use those two trajectories to analyze our simulations. However, the simulation boxes have different number of atoms due to the solvation and ionization procedure in the beginning of simulations. We therefore need to load trajectories with their related structure files.

```
In [4]: dcd1 = Trajectory('initial_trajectory.dcd')

In [5]: dcd2 = Trajectory('final_trajectory.dcd')

In [6]: dcd1
Out[6]: <Trajectory: initial_trajectory (next 0 of 40 frames; 29031 atoms)>

In [7]: dcd2
Out[7]: <Trajectory: final_trajectory (next 0 of 40 frames; 36588 atoms)>

In [8]: structure1 = parsePDB('initial_ionized.pdb')

In [9]: structure2 = parsePDB('final_ionized.pdb')
```

```
In [10]: structure1
Out[10]: <AtomGroup: initial_ionized (29031 atoms)>

In [11]: structure2
Out[11]: <AtomGroup: final_ionized (36588 atoms)>
```

In order to analyze the two trajectories together, we must have the same number of atoms in the sets that we analyze. To do this we link the trajectories to there corresponding initial structures and set the C-alpha atoms as the active ones with following commands:

```
In [12]: dcd1.setCoords(structure1)

In [13]: dcd1.setAtoms(structure1.calpha)

In [14]: dcd1
Out[14]: <Trajectory: initial_trajectory (next 0 of 40 frames; selected 214 of 29031 atoms)>

In [15]: dcd2.setCoords(structure2)

In [16]: dcd2.setAtoms(structure2.calpha)

In [17]: dcd2
Out[17]: <Trajectory: final_trajectory (next 0 of 40 frames; selected 214 of 36588 atoms)>
```

## 5.2 Concatenating Trajectory Files

It is not possible to concatenate Trajectory objects inside Python but must instead be done using files. We therefore write out new files filtered to contain only C-alpha atoms as set in the previous step:

```
In [18]: writeDCD('initial_filtered.dcd', dcd1)
Out[18]: 'initial_filtered.dcd'

In [19]: writeDCD('final_filtered.dcd', dcd2)
Out[19]: 'final_filtered.dcd'
```

One way to combined trajectories into the same object is the `Trajectory.addFile()` method of `Trajectory` objects.

```
In [20]: traj = Trajectory('initial_filtered.dcd')

In [21]: traj.addFile('final_filtered.dcd')
```

Alternatively we can create an `Ensemble` using `parseDCD()`, which gives us the flexibility to do things like reversing the final trajectory to create something we can view in VMD_ rather than having the trajectories both run towards the shared intermediate. We do this as follows:

```
In [22]: combined_traj = parseDCD('initial_filtered.dcd')

In [23]: w2_traj = parseDCD('final_filtered.dcd')

In [24]: for i in reversed(range(len(w2_traj))):
   ....:     combined_traj.addCoordset(w2_traj.getConformation(i))
   ....:

In [25]: combined_traj.superpose()
```

```
In [26]: writeDCD('combined_trajectory.dcd', combined_traj)
Out[26]: 'combined_trajectory.dcd'
```

We also write out a `pdb` file containing just the C-alpha atoms, which can be loaded into VMD_ together with this combined trajectory for visualization.

```
In [27]: writePDB('initial_filtered.pdb', structure1.ca)
Out[27]: 'initial_filtered.pdb'
```

## 5.3 Principal Component Analysis

We next perform PCA on the concatenated trajectory as follows.

```
In [28]: pca = PCA('Adelynate Kinase coMD')

In [29]: pca.buildCovariance(combined_traj)

In [30]: pca.calcModes()
```

The first half of the trajectory is from the initial structure and the second half of the trajectory is from the final structure. We can identify these two trajectories as follows.

```
In [31]: forward = combined_traj[0:40]

In [32]: backward = combined_traj[40:]
```

## 5.4 Visualization of Trajectories

Finally, the trajectories can be plotted by using the `showProjection()` function:

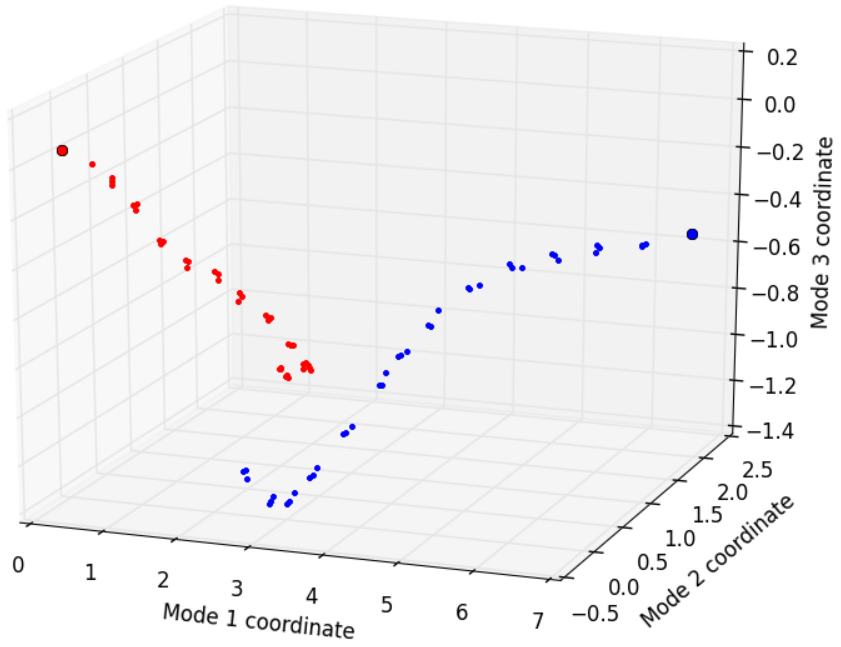
```
In [33]: showProjection(forward, pca[:3], color='red', marker='.');

In [34]: showProjection(backward, pca[:3], color='blue', marker='.');

In [35]: showProjection(forward[0], pca[:3], color='red', marker='o');

In [36]: showProjection(backward[0], pca[:3], color='blue', marker='o');
```

The plots will be in the following form:

Having calculated the modes, we can write them to a `.nmd` file for viewing in NMWiz[9].

```
In [37]: writeNMD('ake_pca.nmd', pca, structure1.ca)
Out[37]: 'ake_pca.nmd'
```

[9] http://prody.csb.pitt.edu/nmwiz/


0.2
0.0
−0.2
−0.4
−0.6
−0.8
−1.0
−1.2
−1.4
Mode 3 coordinate
0
1
2
3
4
5
6
7
Mode 1 coordinate
−0.5
0.0
0.5
1.0
1.5
2.0
2.5
Mode 2 coordinate

CHAPTER

SIX

# LICENSE

```
Copyright (c) 2013, University of Pittsburgh
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:
    * Redistributions of source code must retain the above copyright
      notice, this list of conditions and the following disclaimer.
    * Redistributions in binary form must reproduce the above copyright
      notice, this list of conditions and the following disclaimer in the
      documentation and/or other materials provided with the distribution.
    * Neither the name of the <organization> nor the
      names of its contributors may be used to endorse or promote products
      derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL <COPYRIGHT HOLDER> BE LIABLE FOR ANY
DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
(INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

**Acknowledgments**

Continued development of Protein Dynamics Software *ProDy* and associated programs is partially supported by the NIH[10]-funded R01 GM139297 entitled *"Toward a deeper understanding of allostery and allotargeting by computational approaches"*.

[10] http://www.nih.gov/

[MG13] Gur M, Madura J, Bahar I Global Transitions of Proteins Explored by a Multiscale Hybrid Methodology: Application to Adenylate Kinase[2] *Biophys J* **2013** 7:1643–1652

[2] http://www.sciencedirect.com/science/article/pii/S000634951300934X