POZNAN UNIVERSITY OF TECHNOLOGY

FACULTY OF COMPUTING AND TELECOMMUNICATION
Institute of Computing Science

Bachelor's thesis

# WORKFLOW FOR GENERATION OF PROTEIN ENSEMBLES

Mikołaj Koszczyc, 145180

Supervisor
Jan Brezovsky, PhD

POZNAŃ 2022

# Contents

# Chapter 1

# Introduction

Proteins are known to be dynamic entities. To predict and understand their function, instead of a single static structure, we need to look at the diverse set of their conformations. This complex phenomenon called protein dynamics consists of motions happening with various timescales and amplitudes dependent on the system and the environment. These movements may take form of quick subangstrom vibrations of covalent bonds or slower but more prominent, coordinated movements of several residues in a sequence. Many of these conformational changes involve nontrivial moves spanning the space of several angstroms, requiring several nanoseconds to execute due to synchronization with changes in surrounding residues. At present, due to the high availability of high-quality three-dimensional (3D) protein structures obtained using experimental methods, computational biophysical methods are progressively more applied to study protein motions at atomic resolution. Software like ProDy or PyTraj used to generate and evaluate conformational ensembles is not easy to quickly implement into day-to-day lab work. While thorough, its documentation is extensive and very technical thus creating a high barrier to entry and might discourage scientists from implementing it in their work more often.

## 1.1 Aim and scope of thesis

The aim of this thesis is to create a Python library designed to organize and simplify the workflow for generation and evaluation of protein conformational ensembles, as well as the necessary documentation and the workflow example. A well-documented library might prove to be a useful tool enabling easier and faster generation and analysis of conformational ensembles. Creating diverse ensembles of protein conformations is critical in studying protein solution structures and their functions.

## 1.2 Overview

This thesis is divided such that the essential theory needed to understand the problem of generating conformational ensembles is presented first. The theory and tools described in chapter 2 are important in order to appreciate why the simplifications are made. Chapter 3 describes in greater detail the implementation of software and original work, presents the example workflow for generation of protein ensembles and tests aforementioned workflow on diverse proteins to check its viability.

# Chapter 2

# Theory

## 2.1 Biology of proteins

### 2.1.1 Protein Structure

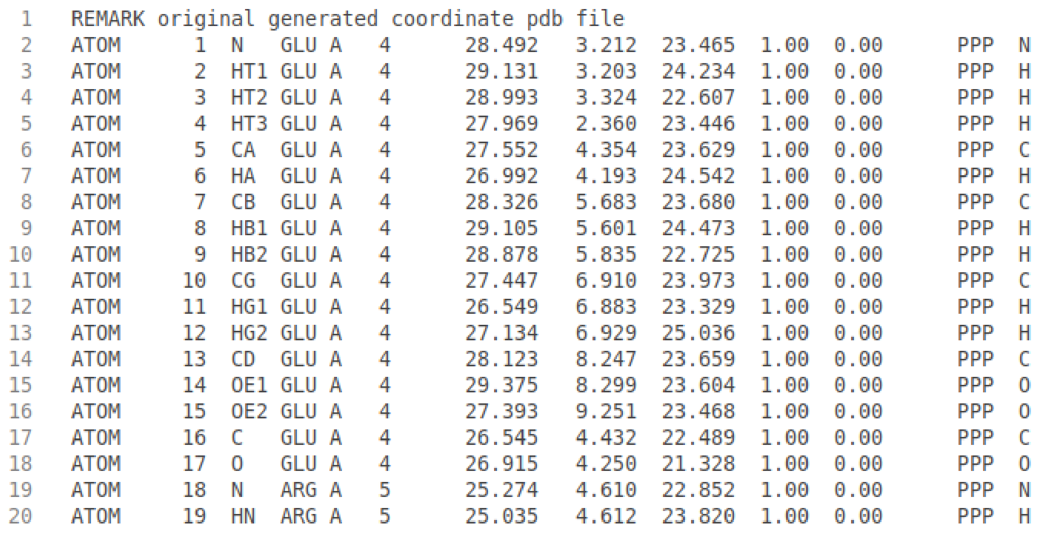
Proteins are highly complex substances which are present in all living organisms. They provide great nutritional value and are directly involved in essential chemical processes. In the early 19th century, their importance was first recognized by scientists, including Swedish chemist Jöns Jacob Berzelius who created the term *protein* originating from the Greek prōteios, meaning "holding first place".[1]
From a chemical point of view, proteins are by far the most structurally complex and functionally sophisticated molecules known.[2] Consisting of many connected amino acids they form long chains, much like beads on the string, they are very large compared to other molecules of sugar or salt.
Proteins are built of amino acids, small organic molecules, which consist of an alpha (central) carbon atom linked to an amino group, a carboxyl group, a hydrogen atom and a variable component called a side chain.[3] Although more than 100 amino acids can be found in nature, only about 20 types occur naturally in proteins.

It is important to note that proteins of similar functions have similar amino acid composition and sequence.[2] Because of that, it is possible to find correlations between protein structure and its function, which may be connected to the properties of its amino acid components.

#### PDB data

The PDB file format was created to store macromolecular data in the RCSB database.
*Protein Data Bank* (*PDB*) includes all the public 3D structures for proteins, nucleic acids and carbohydrates.[4] PDB records contain protein or nucleotide sequence, a short description, biological source of the data and atomic coordinates (x, y, z) of macromolecules.[5]
The 3D structure data in this database largely comes from X-ray crystallography and NMR spectroscopy and provides information on the biological function of the protein and connected mechanisms.

PDB files (Figure 2.1) are long text files which can be read using any text editor, although atom coordinates inside them are not comprehensible by reading. PDB files do not include connectivity data.



```
REMARK original generated coordinate pdb file
ATOM      1  N   GLU A   4      28.492   3.212  23.465  1.00  0.00      PPP  N
ATOM      2  HT1 GLU A   4      29.131   3.203  24.234  1.00  0.00      PPP  H
ATOM      3  HT2 GLU A   4      28.993   3.324  22.607  1.00  0.00      PPP  H
ATOM      4  HT3 GLU A   4      27.969   2.360  23.446  1.00  0.00      PPP  H
ATOM      5  CA  GLU A   4      27.552   4.354  23.629  1.00  0.00      PPP  C
ATOM      6  HA  GLU A   4      26.992   4.193  24.542  1.00  0.00      PPP  H
ATOM      7  CB  GLU A   4      28.326   5.683  23.680  1.00  0.00      PPP  C
ATOM      8  HB1 GLU A   4      29.105   5.601  24.473  1.00  0.00      PPP  H
ATOM      9  HB2 GLU A   4      28.878   5.835  22.725  1.00  0.00      PPP  H
ATOM     10  CG  GLU A   4      27.447   6.910  23.973  1.00  0.00      PPP  C
ATOM     11  HG1 GLU A   4      26.549   6.883  23.329  1.00  0.00      PPP  H
ATOM     12  HG2 GLU A   4      27.134   6.929  25.036  1.00  0.00      PPP  H
ATOM     13  CD  GLU A   4      28.123   8.247  23.659  1.00  0.00      PPP  C
ATOM     14  OE1 GLU A   4      29.375   8.299  23.604  1.00  0.00      PPP  O
ATOM     15  OE2 GLU A   4      27.393   9.251  23.468  1.00  0.00      PPP  O
ATOM     16  C   GLU A   4      26.545   4.432  22.489  1.00  0.00      PPP  C
ATOM     17  O   GLU A   4      26.915   4.250  21.328  1.00  0.00      PPP  O
ATOM     18  N   ARG A   5      25.274   4.610  22.852  1.00  0.00      PPP  N
ATOM     19  HN  ARG A   5      25.035   4.612  23.820  1.00  0.00      PPP  H
```

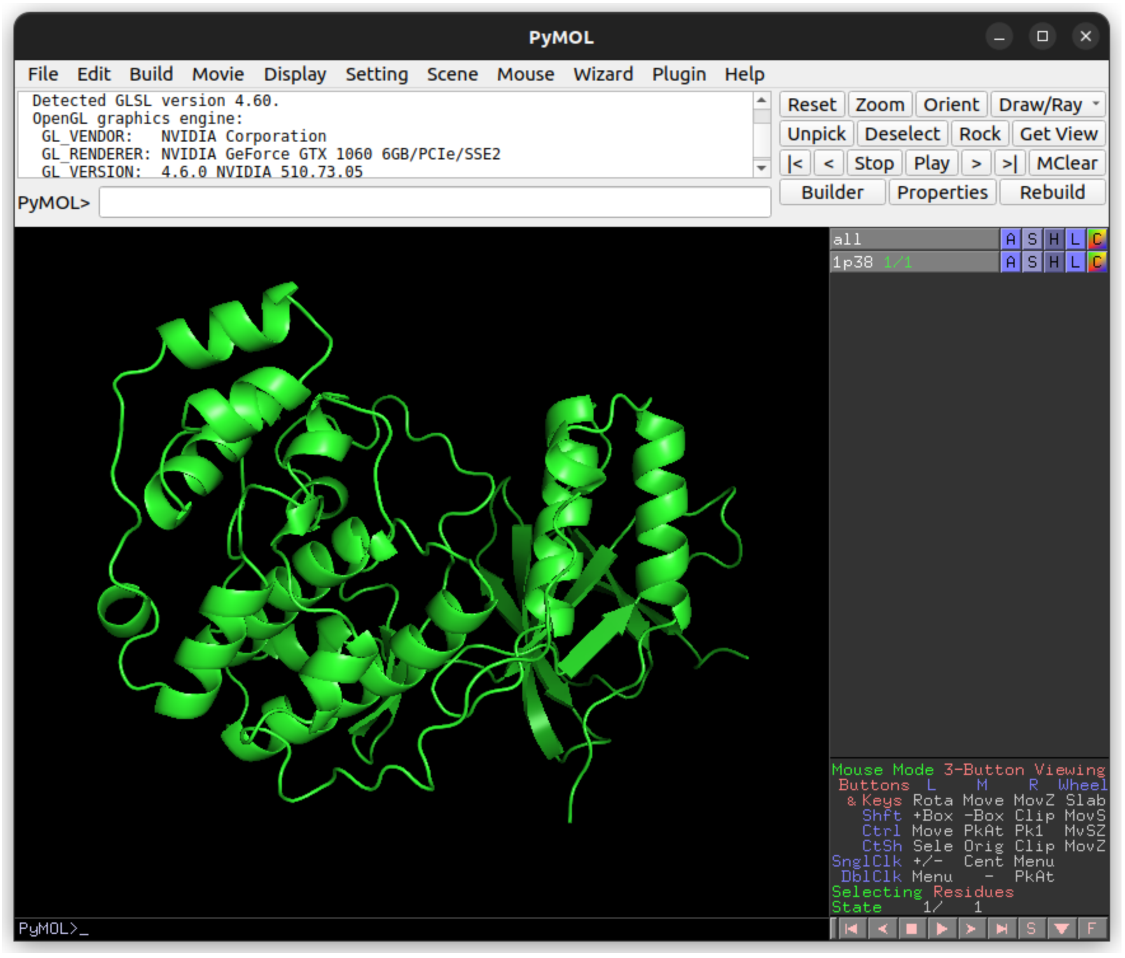
Figure 2.1: Content of the PDB file (PDB-ID: 1P38).

### Visualizing Structures

While PDB files can be viewed directly using any text editor, a visualization program is more convenient for viewing their contents. PDB files can be browsed online on the RCSB PDB website. This website allows users to search for the entries of their interest. Visualization programs such as VMD or PyMOL can be used to read PDB files and visualize the protein structures. These programs include measuring tools useful for analysis and identifying interesting regions. To make protein easier to read in the visualization software protein structures can be represented in different models depending on the information user wants to convey:

Table 2.1: Visualization models and their potential use.

| | |
|---|---|
| wire models | comparison |
| ribbon models | highlighting secondary structures |
| ball and stick models | detail |
| surface models | electrostatic potentials |


PyMOL
File Edit Build Movie Display Setting Scene Mouse Wizard Plugin Help
Reset Zoom Orient Draw/Ray
Unpick Deselect Rock Get View
|< < Stop Play > >| MClear
Builder Properties Rebuild
PyMOL>
all A S H L C
1p38 1/1 A S H L C
PyMOL>_

```
Detected GLSL version 4.60.
OpenGL graphics engine:
 GL_VENDOR:   NVIDIA Corporation
 GL_RENDERER: NVIDIA GeForce GTX 1060 6GB/PCIe/SSE2
 GL_VERSION:  4.6.0 NVIDIA 510.73.05
```

```
Mouse Mode 3-Button Viewing
 Buttons  L    M    R  Wheel
  & Keys Rota Move MovZ Slab
    Shft +Box -Box Clip MovS
    Ctrl Move PkAt Pk1  MvSZ
    CtSh Sele Orig Clip MovZ
 SnglClk +/-  Cent Menu
  DblClk Menu  -   PkAt
Selecting Residues
State     1/   1
```

Figure 2.2: PyMOL visualization of protein structure (green cartoon) (PDB-ID: 1P38).

### 2.1.2 Protein Dynamics

Proteins can lack a stable tertiary structure, which makes them flexible. In addition to unfolded and naturally disordered proteins, many folded proteins display large movements between conformational ensembles. These motions can be connected with protein functional cycles such as reactions in enzymes.[6] Because of that their shape cannot be well described using static structure, but rather as an ensemble of conformations. Its calculation is reliant on experimental measurements, primarily *Nuclear Magnetic Resonance* (*NMR*) and X-ray crystallography. Structural biologists can use molecular dynamics simulations to generate conformational ensembles where conformational sampling can be controlled using a set of rules.[7]

## 2.2 Conformational Ensembles

Protein conformational ensembles represent the range of conformations sampled at the state of protein equilibrium.[8]
Conformational ensembles are useful tools to explore the basic properties of proteins like the early stages of protein folding, the mechanisms of molecular recognition and the means of propagating information through the correlated movement of backbone residues.

## 2.3 Generating Conformational Ensembles

### 2.3.1 Molecular dynamics (MD)

*Molecular dynamics* (*MD*)[9] is a simulation method providing an exceptionally detailed description of protein dynamics. To achieve this, it extracts information about the amplitude of protein dynamics from the experimental data and relies on modelling interatomic interactions with the use of force fields (empiric potentials). The final accuracy of the trajectories depends on the quality of these force fields and on the computer hardware used for calculations.[8]

Although *graphics processing units* (*GPUS*) are now primarily designed and used for gaming or mining cryptocurrencies, they perform MD very efficiently. In recent years the computational power of this kind of hardware has drastically improved which enabled this kind of simulation even for modest experimental labs.[10]

In cases where the scope of the system and the timescale of the dynamics allow for an MD analysis, this is a very valuable technique for describing the fluctuations of the protein structure. It is however necessary to validate the resulting trajectories by either predicting NMR parameters or by predicting the result of perturbations of the system such as point mutations.[8] These validations are needed to consider the MD trajectories as a realistic behaviour of the protein.[11]

## 2.4 Elastic Network Models

*Elastic Network Models* (*ENM*) are the simplified models of proteins used to study slow structural dynamics. ENMs are made of point-like particles and linear springs connecting them. These models are suitable for *Normal Mode Analysis* (*NMA*) around a given reference structure. The use of particles instead of all-atom models significantly lowers the computational costs and is also convenient for comparative studies between multiple protein structures.

The connectivity is defined by calculating the distance between each pair of particles in the reference structure. If the calculated value is smaller than the specified cutoff distance, two particles are connected by a spring with the same stiffness constant. The equilibrium length of each spring is set equal to the distance between the particles in the reference structure so that the reference structure corresponds to the most stable state.[6]

In the original ENM for proteins, each atom was represented by a particle. To lower computational costs, coarse-grained models are more broadly used. In coarse-grained models, each amino acid residue is represented by a particle significantly lowering the number of particles. When creating coarse-grained ENM, each $\alpha$-carbon atom in the reference structure is replaced with a particle and the cutoff distance is set around 10 Å.

ENMs are commonly represented and analyzed in one of two ways: *Anisotropic Network Models* (*ANM*) and *Gaussian Network Models* (*GNM*).

### 2.4.1 Anisotropic Network Model (ANM)

*Anisotropic Network Models* (*ANM*) are the most common ENMs. In these network models the coordinates are described by the 3N mass-weighted coordinates of the nodes[12]:

$$r = (\Delta x_1, \Delta y_1, \Delta z_1, ..., \Delta x_N, \Delta y_N, \Delta z_N)^T \text{, where } \Delta x_i = x_i - x_i^0 \tag{2.1}$$

is the $x$-component of the movement of node $i$ from its equilibrium position, $r_i^0$. Interactions are stored in the $3N \times 3N$ Hessian matrix. The Hessian matrix can be understood as an $N \times N$ matrix of $3 \times 3$ submatrices. Each submatrix characterizes the energetic input from the interaction between two nodes.
Using the notation $x_{ij} = (x_j - x_i)$ and adequately for $y_{ij}$ and $z_{ij}$,
the elements beyond the diagonal are:

$$H_{ij} = -\frac{\gamma_{ij}}{R_{ij}^2} \begin{bmatrix} x_{ij}^2 & x_{ij}y_{ij} & x_{ij}z_{ij} \\ x_{ij}y_{ij} & y_{ij}^2 & y_{ij}z_{ij} \\ x_{ij}z_{ij} & y_{ij}z_{ij} & z_{ij}^2 \end{bmatrix}, \tag{2.2}$$

where $\gamma_{ij}$ is the spring constant between nodes i and j,
and $R_{ij}$ is the equilibrium distance between these two nodes.
The elements on the diagonal satisfy:

$$H_{ij} = \sum_{j;j \neq i} H_{ij} \tag{2.3}$$

The common practice is to set the identical spring constant $\gamma_{ij} = \gamma$ to all pairs of nodes where the distance between them is smaller than the set cutoff distance. Spring constants for all other nodes are set to 0. Empirical studies[13] found that using $\alpha$-carbons of a protein as the nodes and setting the cutoff distance to 15Å generate *Root-Mean-Square Fluctuations* (*RMSF*) that correlate with experimental B-factors.

### 2.4.2 Gaussian Network Model (GNM)

*Gaussian Network Models* (*GNM*) are a simplification of ANM models described above.[14] GNM assume the isotropic and Gaussian form of the fluctuations. As a result, the interaction matrix can be reduced from $3N \times 3N$ Hessian to $N \times N$ Kirchhoff matrix.
Due to the fundamental assumption of vibrational isotropy, GNM is able to predict the extent of motions together with their cross-correlation, although it cannot define their directions. It additionally enables us to find the domains involved in joint motions in the global modes, but not the direction of such rearrangements.
GNM uses the elastic constant $\gamma$ as its only parameter. $\gamma$ is used to define actions between nodes which are less than a cutoff distance apart.

### 2.4.3 Normal Mode Analysis (NMA)

*Normal Mode Analysis* (*NMA*) is used to investigate structural transitions or fluctuations in ENMs.[6] NMA provides information on the equilibrium modes accessible to a system. This is a general method, widely used in molecular mechanics research together with all-atom models.

## 2.5 Evaluating Conformational Ensembles

### 2.5.1 RMSD

In bioinformatics, the *Root-Mean-Square Deviation* (*RMSD*) is the measurement of the average distance between the atoms of superimposed proteins. This measure can be used to quantify the similarity between reference and generated structures.

$$RMSD(a,b) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(a_{ix}-b_{ix})^2+(a_{iy}-b_{iy})^2+(a_{iz}-b_{iz})^2}$$

where $a_i$ and $b_i$ refer to atoms in each molecule with subscripts marking x/y/z coordinates respectively.

In the study of protein conformations, the similarity between 3D structures is measured using RMSD of $\alpha$-carbons in the protein. RMSD of zero means that two structures are identical.

### 2.5.2 RMSF

*Root-Mean-Square Fluctuation* (*RMSF*) is a measurement analogous to RMSD. RMSF is a calculation of individual residue flexibility. For analysis of protein structures it is usually restricted to backbone or $\alpha$-carbon atoms because these residues are more characteristic of conformational changes than side-chains. RMSF per residue is plotted against residue number and can be used to visualize most mobile parts of the protein.

### 2.5.3 Hydrogen Bonds

*Hydrogen bonds* (*H-BONDS*) are vital for most of the directional interactions that support protein structure. H-bonds are formed by the interaction of a hydrogen atom bonded to an electronegative atom (donor) with another electronegative atom (acceptor). The frequently observed distance for H-bonds is lower than 2.5Å between donor and the acceptor with a donor-hydrogen-acceptor- angle between 90°and 180°. Most of protein structures have their core composed of secondary structures such as $\alpha$-helices and $\beta$-sheets [15]. These types of structures create most stable conformations of polypeptide chains by maximizing the hydrogen-bonding potential.

### 2.5.4 Secondary Structure

Protein structures are also classified by their secondary structure.[16] Secondary structure is a regular, local structure of protein backbone, stabilized using hydrogen bonds.

#### Alpha helices

The *alpha helix* (*$\alpha$-HELIX*) has a right-handed spiral conformation. In this structure, every backbone amine group creates a H-bond with the carboxylic group of amino acid four residues later in the sequence.

#### Beta strands

The *beta strand* (*$\beta$-STRAND*) is a 3-10 amino acid long chain with almost completely extended conformation. Multiple parallel or anti-parallel adjacent beta strands create a structure called *beta sheet* (*$\beta$-SHEET*)

# Chapter 3

# Software

## 3.1 Tools used

### 3.1.1 Python

Python is a programming language widely used in the development of bioinformatics tools. It is a high-level, interpreted, object-oriented, and high-level-typed language. Python is designed to be highly readable and easily extensible. [17]

This workflow uses Python 3.7 as its primary language. Although the newer version of this language is available, version 3.7 is the recommended version for this workflow. Other versions of Python are not supported due to compatibility issues with other dependencies.

### 3.1.2 Python Package Index (PyPI)

The Python Package Index (PyPi) is a Python package repository. It is a web-based repository where users can upload and manage Python packages. PyPi is the most popular Python package repository on the internet. This tool is used for installing Python packages required for this workflow.

### 3.1.3 Anaconda

Anaconda is a Python distribution that includes a wide range of tools for data science. It simplifies package management and acts as a robust environment. Anaconda or its lightweight alternative, Miniconda is the recommended Python distribution for this workflow.

### 3.1.4 ProDy

ProDy is a free and open-source Python package for protein structural dynamics analysis.[18] It is an integrated application programming interface (API) created for protein dynamics modelling and analysis. This tool lays a foundation for this workflow.
It is available free of charge under MIT licence from `https://github.com/prody/ProDy`

### 3.1.5 PYTRAJ

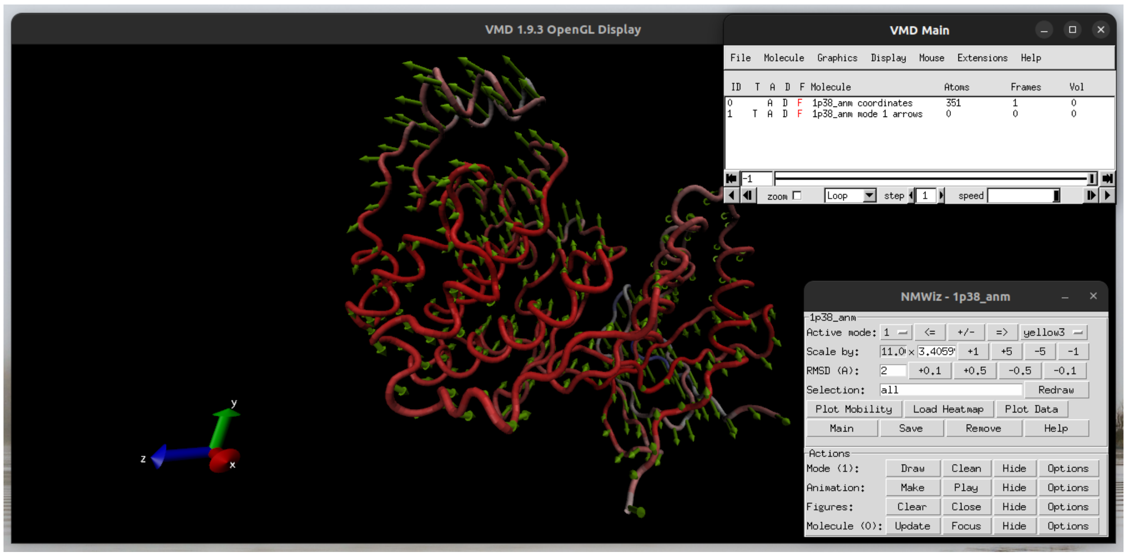
PYTRAJ is a Python package for trajectory analysis. It is a fast and versatile trajectory analysis package, supporting more than 80 types of data analyses (rmsd, rmsf, radgyr etc.). [19]
PYTRAJ is a part of AMBER suite distribution but can be used independently.

### 3.1.6 NAMD

NAMD is a parallel molecular dynamics code. It is designed for high-performance simulation of biomolecular systems. This tool uses Charm++ parallel objects to be able to scale up to even 500,000 cores.[10] NAMD enables robust parallel computation using supercomputers but can be scaled down for even modest experimental labs.
It can utilize multiple GPUs for faster computation, a valuable feature for this workflow as it enables smaller budget teams to work with MD.

### 3.1.7 VMD

VMD (Visual Molecular Dynamics) is a program for molecular visualization. It allows users to view 3D protein structures and trajectories, create animations and perform analyses. It supports PDB file format as well as scripting.
The alternative to VMD is PyMOL, although VMD is used for this workflow.



```
ID  T  A  D  F  Molecule                    Atoms   Frames   Vol
0      A  D  F  1p38_anm coordinates        351     1        0
1   T  A  D  F  1p38_anm mode 1 arrows      0       0        0
```

```
-1        Loop    step 1

1p38_anm
Active mode:  1     <=   +/-   =>   yellow3
Scale by:     11.0 x 3.4059   +1   +5   -5   -1
RMSD (A):     2     +0.1   +0.5   -0.5   -0.1
Selection:    all
Mode (1):
Animation:
Figures:
Molecule (0):
```

Figure 3.1: VMD visualization of protein structure (tubes and trajectory arrows) (PDB-ID: 1P38).

## 3.2 Implementation

This part describes the workflow for the generation and analysis of protein conformational ensembles.
Steps are intended to be used in the order in which they are described.

### 3.2.1 Loading data

The workflow begins by loading protein data from the PDB file. It can be achieved using *load_pdb()* method, which takes the four-letter accession code for the PDB (pdb_id) as its input. The PDB file in the working directory (the directory from which the script is invoked) takes priority. If such a file does not exist, it is downloaded from the RCSB protein database.

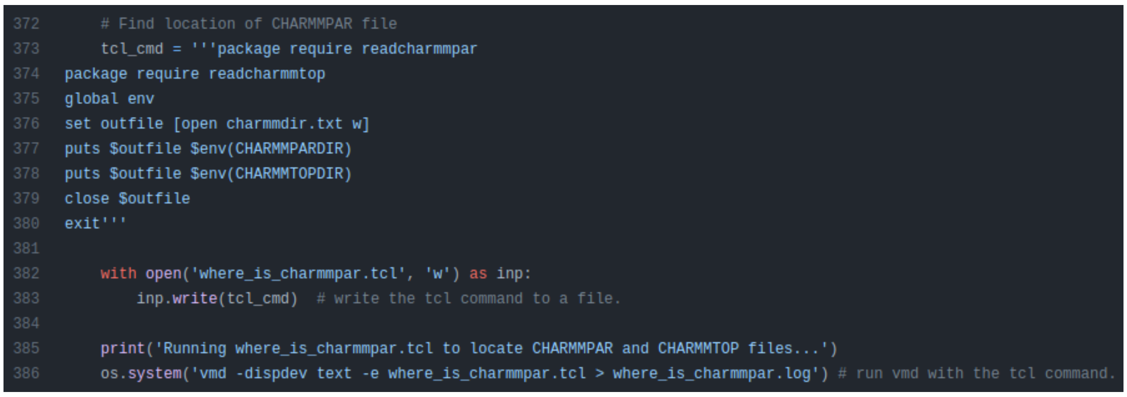
Contents of the PDB file are stored in the dictionary called *proteins[]* under the key equal to the pdb_id. This solution helps easily store and navigate the elements created later in the workflow.

### 3.2.2 Selecting protein

To ensure the correctness of future calculations, all additional atoms are removed from the PDB using *remove_waters()* method. This method takes the same pdb_id as its input. The removal is done using the custom tcl script which is run using VMD. It selects only the protein structure and overwrites the PDB file. For debugging purposes, the log file is created containing all output from the VMD. The new PDB file is loaded into the *proteins[]* dictionary.

### 3.2.3 Generating PSF

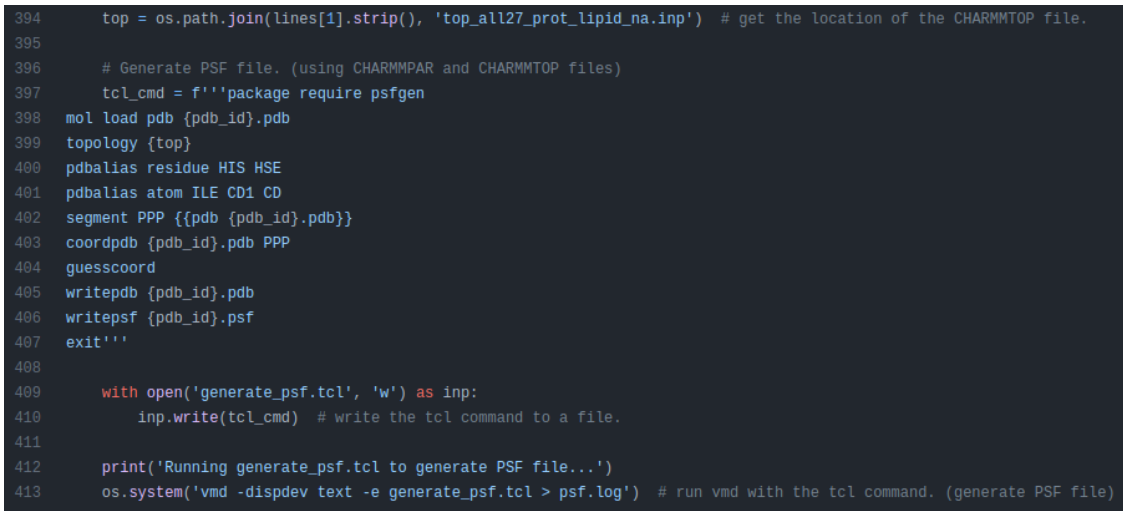
The protein structure file (PSF) contains the molecule-specific information needed to optimize protein conformations using NAMD. NAMD and VMD require the PSF to precisely match the order of atoms in the PDB to work correctly. Generating a new PSF (Figure 3.3) requires the use of an existing CHARMM topology file whose location is acquired from VMD using the tcl script.



```
    # Find location of CHARMMPAR file
    tcl_cmd = '''package require readcharmmpar
package require readcharmmtop
global env
set outfile [open charmmdir.txt w]
puts $outfile $env(CHARMMPARDIR)
puts $outfile $env(CHARMMTOPDIR)
close $outfile
exit'''

    with open('where_is_charmmpar.tcl', 'w') as inp:
        inp.write(tcl_cmd)  # write the tcl command to a file.

    print('Running where_is_charmmpar.tcl to locate CHARMMPAR and CHARMMTOP files...')
    os.system('vmd -dispdev text -e where_is_charmmpar.tcl > where_is_charmmpar.log') # run vmd with the tcl command.
```

Figure 3.2: tcl script responsible for locating CHARMM topology (CHARMMTOP) file.

The PDB file (pdb) and CHARMM topology file (top) are used to write a new PSF by executing another tcl script. The PSF is written in the current working directory. This process can produce

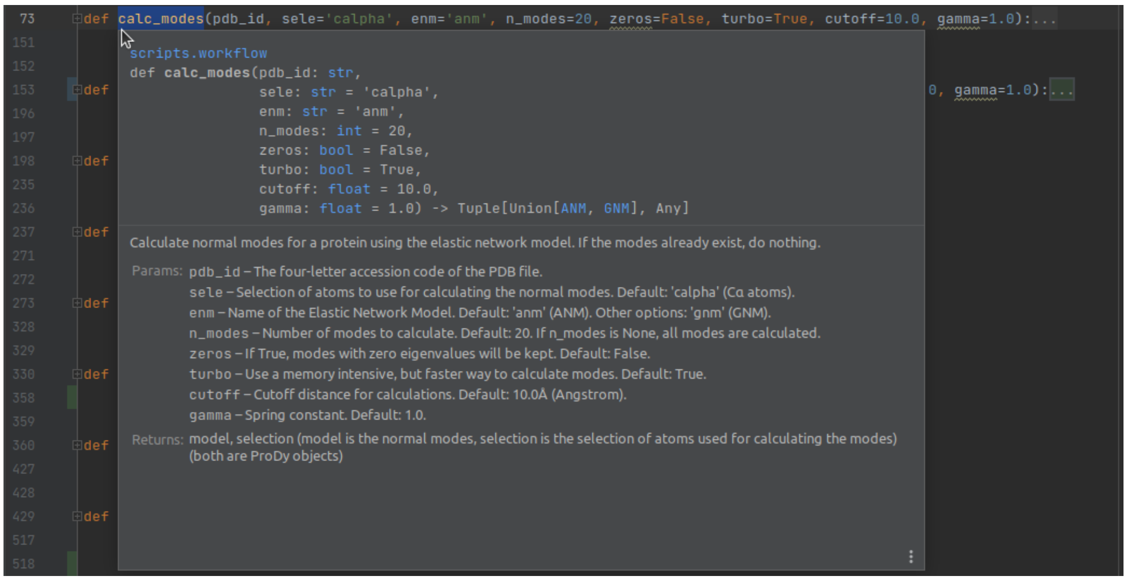


```
    top = os.path.join(lines[1].strip(), 'top_all27_prot_lipid_na.inp')  # get the location of the CHARMMTOP file.

    # Generate PSF file. (using CHARMMPAR and CHARMMTOP files)
    tcl_cmd = f'''package require psfgen
mol load pdb {pdb_id}.pdb
topology {top}
pdbalias residue HIS HSE
pdbalias atom ILE CD1 CD
segment PPP {{pdb {pdb_id}.pdb}}
coordpdb {pdb_id}.pdb PPP
guesscoord
writepdb {pdb_id}.pdb
writepsf {pdb_id}.psf
exit'''

    with open('generate_psf.tcl', 'w') as inp:
        inp.write(tcl_cmd)  # write the tcl command to a file.

    print('Running generate_psf.tcl to generate PSF file...')
    os.system('vmd -dispdev text -e generate_psf.tcl > psf.log')  # run vmd with the tcl command. (generate PSF file)
```

Figure 3.3: tcl script responsible for generating PSF.

errors due to incorrect residue names in the input PDB file. Thus the log file is created and checked for any issues before continuing the workflow. Issues with PDB files can be very specific and should not be generalized for this workflow. The user is expected to resolve these issues by themself using provided log files.

### 3.2.4 Normal Modes



```
def calc_modes(pdb_id, sele='calpha', enm='anm', n_modes=20, zeros=False, turbo=True, cutoff=10.0, gamma=1.0):...
```

```
scripts.workflow
def calc_modes(pdb_id: str,
               sele: str = 'calpha',
               enm: str = 'anm',
               n_modes: int = 20,
               zeros: bool = False,
               turbo: bool = True,
               cutoff: float = 10.0,
               gamma: float = 1.0) -> Tuple[Union[ANM, GNM], Any]
```

Calculate normal modes for a protein using the elastic network model. If the modes already exist, do nothing.

Params: `pdb_id` – The four-letter accession code of the PDB file.
`sele` – Selection of atoms to use for calculating the normal modes. Default: 'calpha' (Cα atoms).
`enm` – Name of the Elastic Network Model. Default: 'anm' (ANM). Other options: 'gnm' (GNM).
`n_modes` – Number of modes to calculate. Default: 20. If n_modes is None, all modes are calculated.
`zeros` – If True, modes with zero eigenvalues will be kept. Default: False.
`turbo` – Use a memory intensive, but faster way to calculate modes. Default: True.
`cutoff` – Cutoff distance for calculations. Default: 10.0Å (Angstrom).
`gamma` – Spring constant. Default: 1.0.

Returns: model, selection (model is the normal modes, selection is the selection of atoms used for calculating the modes) (both are ProDy objects)

Figure 3.4: Python docstring documentation for the *calc_modes()* method.

#### Calculating normal modes

Method *calc_modes()* compiles all the necessary methods for normal mode calculation. Depending on the input parameters it creates the right ANM or GNM object. Then a proper matrix is built, Hessian or Kirchhoff based on chosen ENM. Users can define values used in this process such as cutoff distance or the spring constant (gamma).
The normal modes for the selection are calculated using the ProDy method called *calcModes()*. At this step, users can choose the number of modes to calculate and the memory intensity of the algorithm (turbo).
Calculated normal modes are saved to a file in .npz format and stored in *proteins[]* dictionary under the key equal to the pdb_id followed by "_anm" (or "_gnm").
The model object and selected atoms are returned as the result and are used in later stages.

If GNM is the ENM of choice, additional graphs are shown. These graphs represent the **contact map** (Fig. 3.5), **cross-correlations** (Fig. 3.6), **slow mode shape** (Fig. 3.7), **square fluctuations** (Fig. 3.8) and **protein structure bipartition** (Fig. 3.9). Graphs are displayed to the user and saved as PNG image files.

#### Extending models

The model obtained in the previous step is created for only selected residues (by default alpha carbons) which means it is coarse-grained and should be extended to all atoms in the protein. Method *extend_model()* is used for this purpose. If normal modes are not present, *extend_model()* can invoke *calc_modes()* to resolve this issue.

The extended model has the same number of modes but uses all atoms instead of only selected nodes. The shape of the mobility plot is analogous. (Fig. 3.11 and 3.12)

The extended model and selected atoms are saved to files, as well as returned by this function.

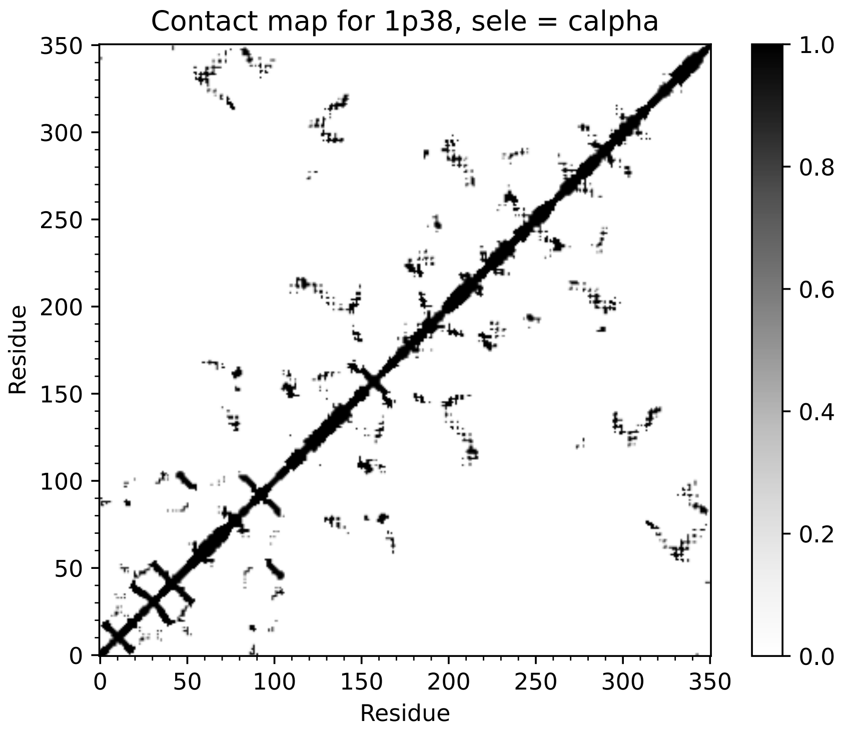


Figure 3.5: Graph representing the contact map for alpha carbons in the protein (PDB-ID: 1P38).

### 3.2.5 Conformational sampling

This section describes the workflow for sampling conformations along ANM modes. Conformation sampling is achieved by the *sample_conformations()* method. Users can declare the number of conformations to sample and the RMSD from the initial conformation.
ANM modes are required for this step. GNM cannot be used.

#### Mode sampling

Provided that prior stages were performed without errors, the ANM model is being sampled using ProDy's *sampleModes()*. As a result, the ensemble of conformations is obtained. The ensemble contains the selected number of conformations and is saved to the file in .dcd format for visualization in VMD.

#### Writing conformations

Function *write_conformations()* is used to save conformations as the ensemble.

New conformations are added to the initial protein structure. Beta values of alpha carbon atoms are set to 1. Other atoms have their beta values set to 0. This action is performed for future optimization of the ensemble. The next stage will use these beta values to optimize atom positions with a harmonic constraint on atoms with beta values set to 1. The goal of this optimization is to refine the covalent geometry of atoms. Maintaining the new alpha carbon atoms position helps to keep the diversity of the ensemble.

Conformations can be found in the new folder created in the working directory. The folder's name is a PDB-ID followed by the "_ensemble"3.15.

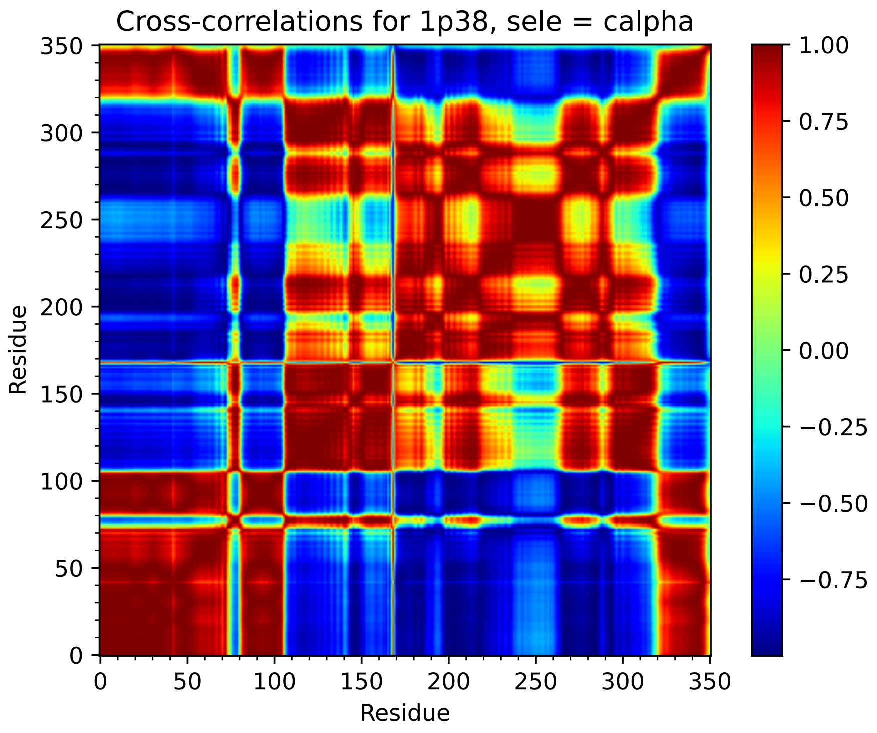


Figure 3.6: Graph representing cross-correlation for alpha carbons in the protein (PDB-ID: 1P38).

### 3.2.6 NAMD optimization

This stage aims to optimize the geometries of conformations using NAMD software.

The conformations used for this stage are generated in previous steps. The method called *optimize_conformations()*(Figure 3.16) is used for optimizing the conformations. It consists of two parts: Locating the NAMD software and using it to optimize the conformations.

#### Locating NAMD

First, the NAMD executable is located using *which()* function which is one of ProDy's utilities. The prefered version of NAMD software is NAMD3. It is remarkably faster than NAMD3 and can utilize GPU acceleration.(Figure 3.17) If NAMD3 isn't available, NAMD2 will be used. If neither is available the error message will be shown asking to install the necessary software.

#### Optimization

Optimization begins with locating the CHARMM files using the *charmmdir.txt* file created during the generation of PSF. Then the NAMD configuration file is generated for each conformation based on the template. (Figure 3.18) User can define some parameters: timestep, cutoff, temperature and the number of steps.

The *multiprocessing* module enables the parallelization of NAMD calculations across multiple CPU cores. The number of CPU cores can be changed by the user depending on the workstation used.

When the NAMD optimization is performed for each conformation in the ensemble directory.(Figure 3.19) The user is notified whether the optimization was finished successfully.
All NAMD output is located in the *_optimize* folder.

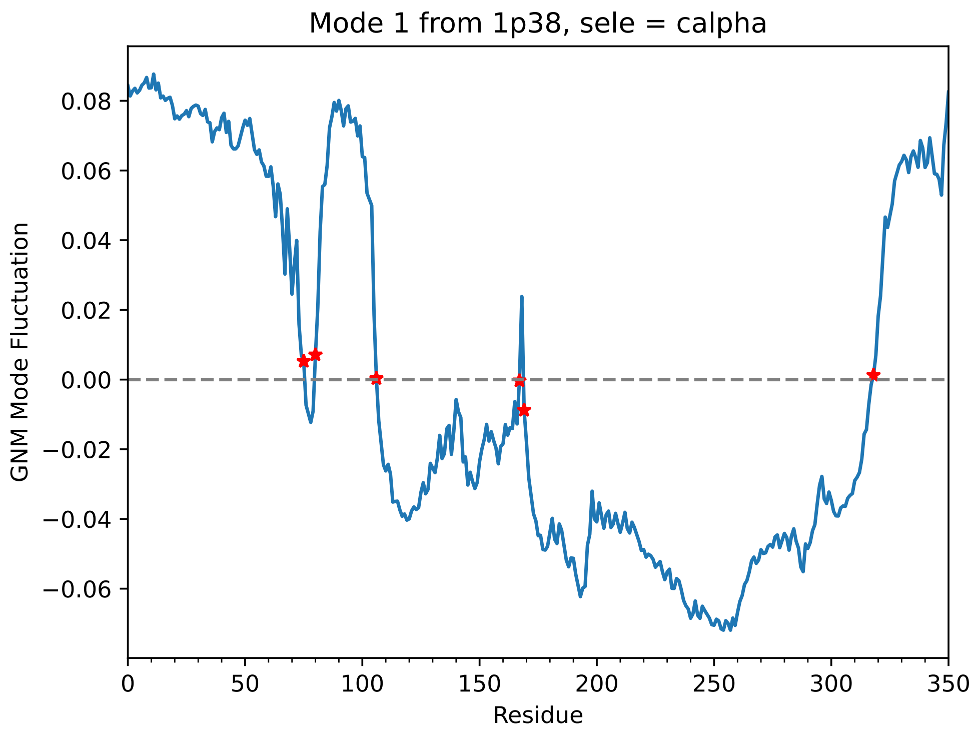


Figure 3.7: Graph representing the slow mode shape in the protein (PDB-ID: 1P38). Red stars indicate hinge sites.

### 3.2.7 Analysis

The last stage of generating diverse protein conformational ensembles is the analysis. The main analyzed attribute is the RMSD change after NAMD refinement.(Figure 3.20) The RMSD change is presented as a graph and saved in the working directory.

The average RMSD of each conformation relative to all others is calculated.(Figure 3.21) This allows picking a diverse set of refined conformations. The indices of the conformations above the set threshold are returned on the output of this method and can be used in scripting. Selected conformations are copied to the new directory.

### 3.2.8 Analyze using PYTRAJ

This step utilizes PYTRAJ to present metrics useful for describing the ensemble. The method called *analyze_pytraj()* requires DCD and PSF files to work correctly. This method returns:

- RMSF and appropriate graph (Figure 3.22)
- Graph for RMSD - Radius of gyration correlation (Figure 3.23)

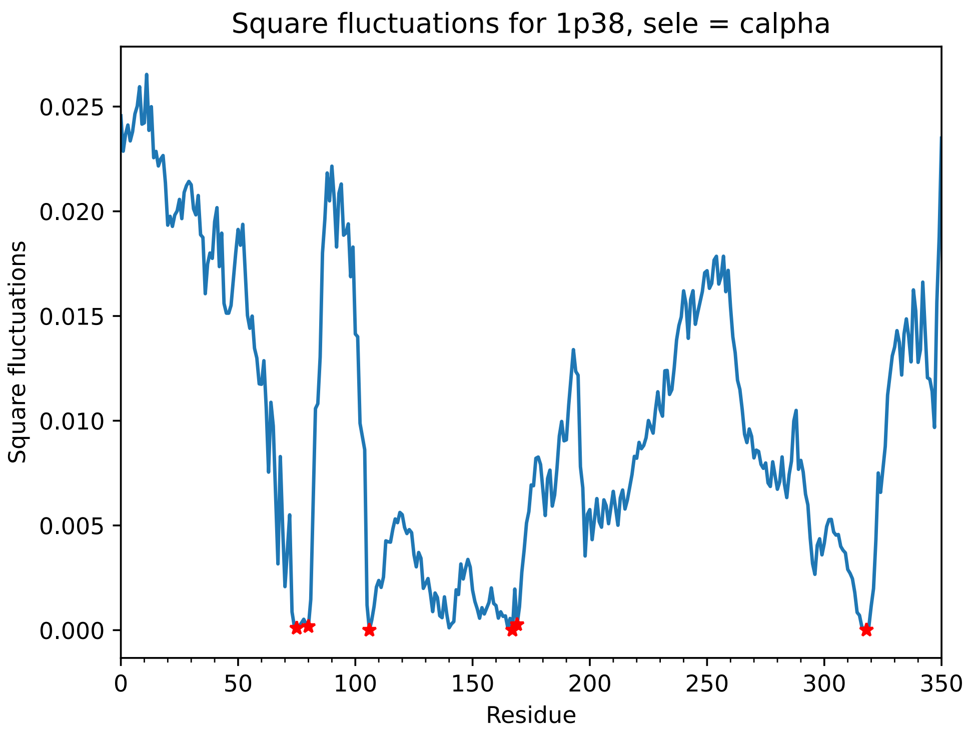


Figure 3.8: Graph representing the square fluctuations (GNM) for alpha carbons in the protein (PDB-ID: 1P38). Red stars indicate hinge sites.

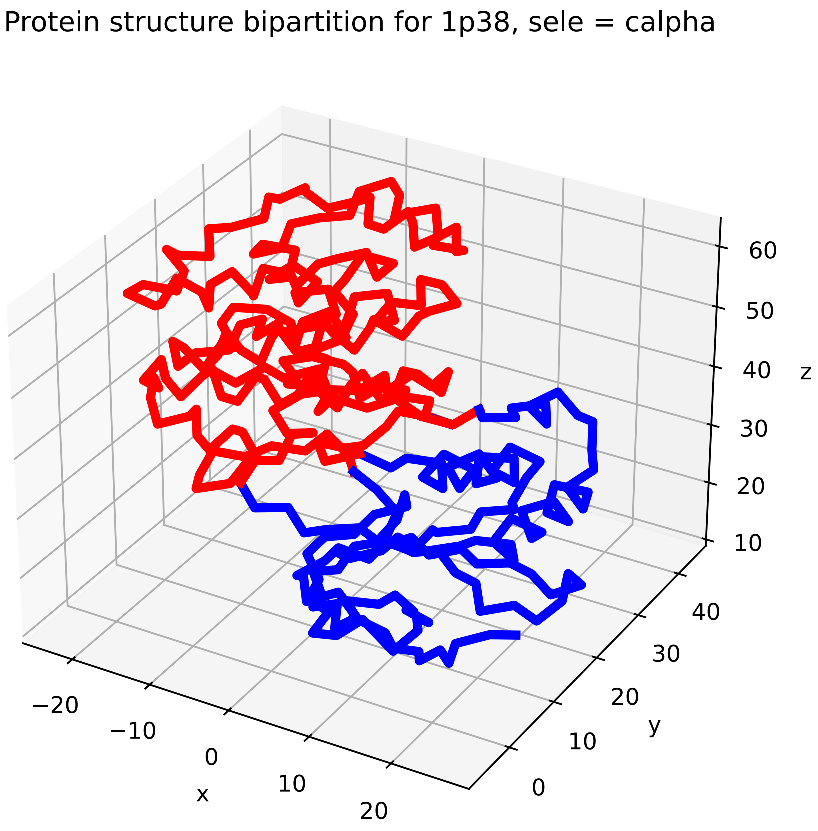
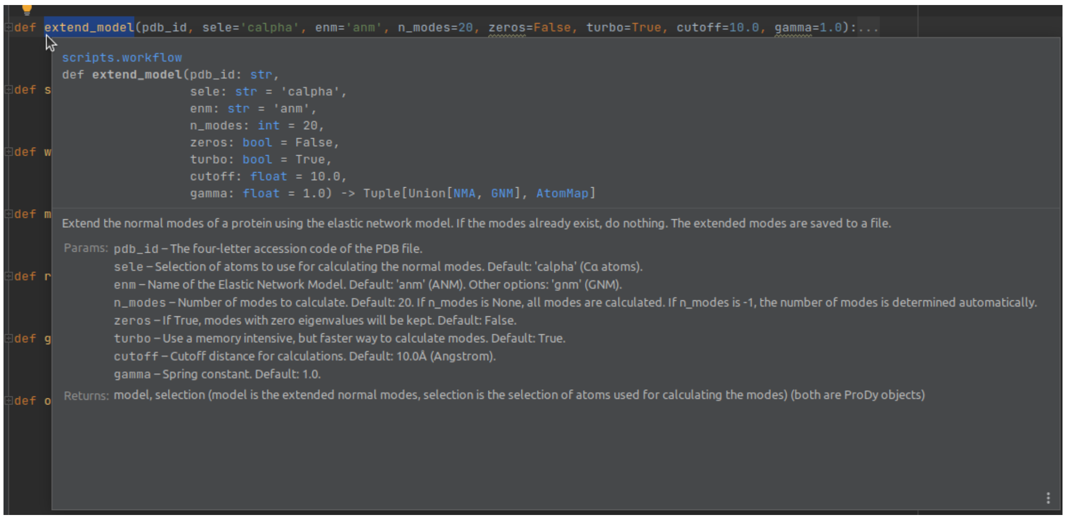


Figure 3.9: Graph representing the protein structure bipartition (PDB-ID: 1P38).



```
def extend_model(pdb_id, sele='calpha', enm='anm', n_modes=20, zeros=False, turbo=True, cutoff=10.0, gamma=1.0):...
def s
def w
def m
def r
def g
def o
```

```
scripts.workflow
def extend_model(pdb_id: str,
                 sele: str = 'calpha',
                 enm: str = 'anm',
                 n_modes: int = 20,
                 zeros: bool = False,
                 turbo: bool = True,
                 cutoff: float = 10.0,
                 gamma: float = 1.0) -> Tuple[Union[NMA, GNM], AtomMap]
```

Extend the normal modes of a protein using the elastic network model. If the modes already exist, do nothing. The extended modes are saved to a file.

Params: `pdb_id` – The four-letter accession code of the PDB file.
`sele` – Selection of atoms to use for calculating the normal modes. Default: 'calpha' (Cα atoms).
`enm` – Name of the Elastic Network Model. Default: 'anm' (ANM). Other options: 'gnm' (GNM).
`n_modes` – Number of modes to calculate. Default: 20. If n_modes is None, all modes are calculated. If n_modes is -1, the number of modes is determined automatically.
`zeros` – If True, modes with zero eigenvalues will be kept. Default: False.
`turbo` – Use a memory intensive, but faster way to calculate modes. Default: True.
`cutoff` – Cutoff distance for calculations. Default: 10.0Å (Angstrom).
`gamma` – Spring constant. Default: 1.0.

Returns: model, selection (model is the extended normal modes, selection is the selection of atoms used for calculating the modes) (both are ProDy objects)

Figure 3.10: Python docstring documentation for the *extend_model()* method.

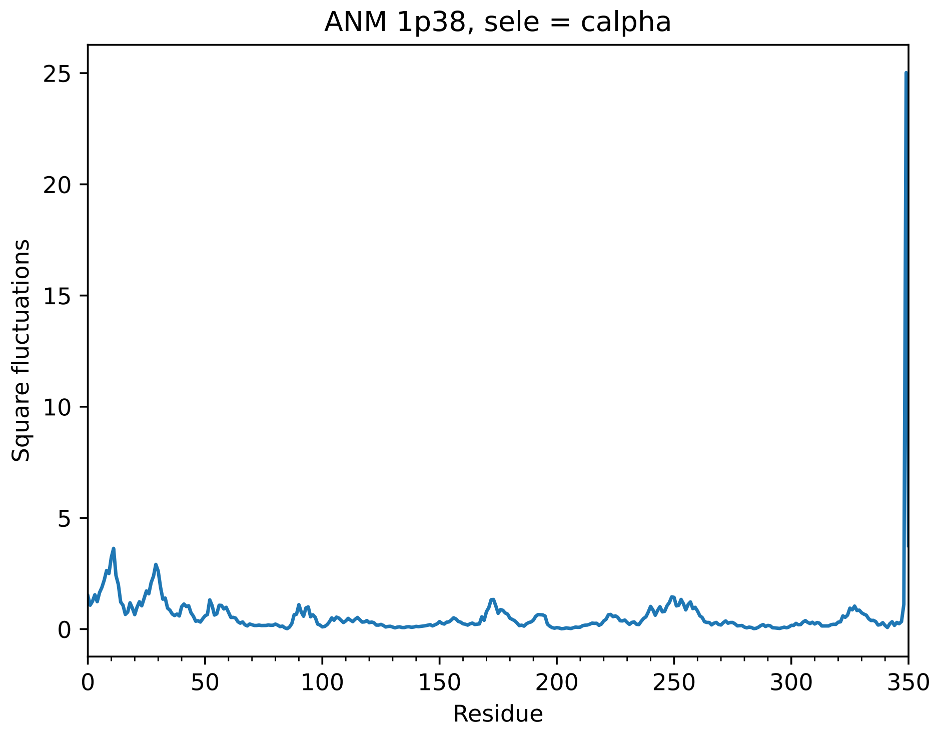


Figure 3.11: Graph representing the square fluctuations (ANM) for alpha carbons in the protein (PDB-ID: 1P38).

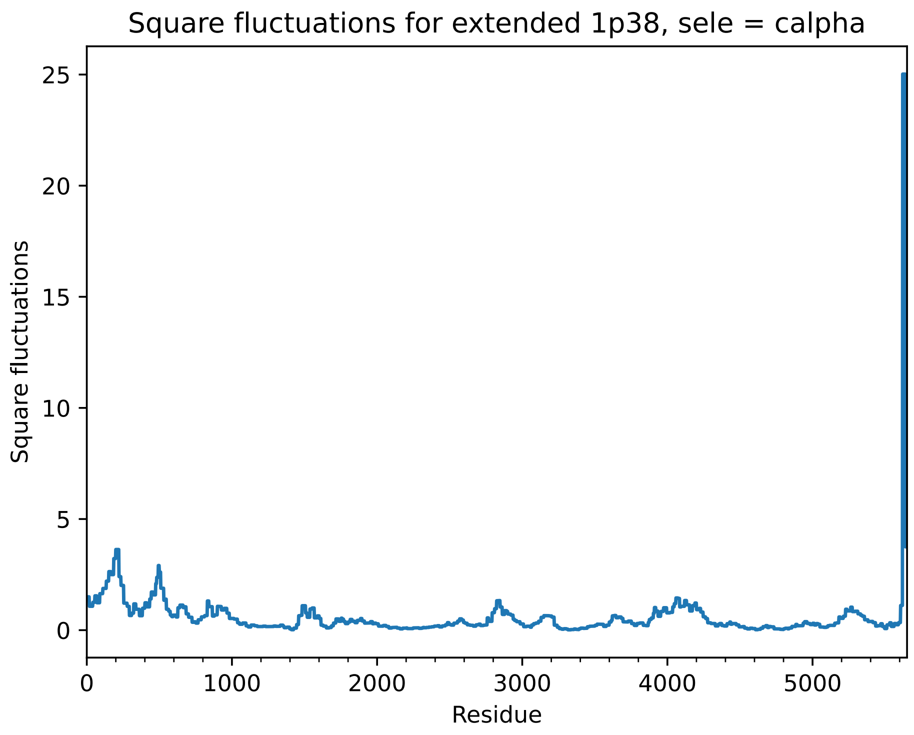
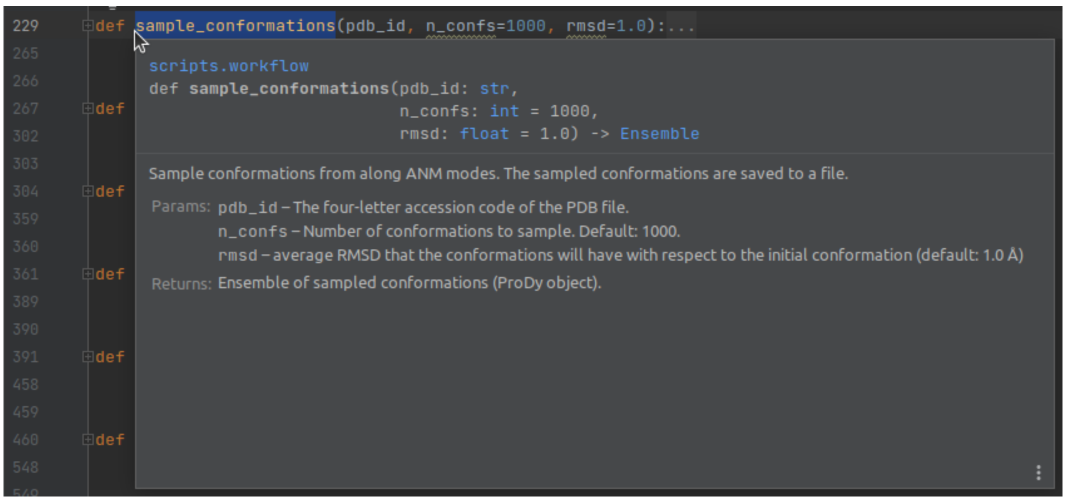


Figure 3.12: Graph representing the square fluctuations in the extended model (PDB-ID: 1P38).



```
def sample_conformations(pdb_id, n_confs=1000, rmsd=1.0):...
```

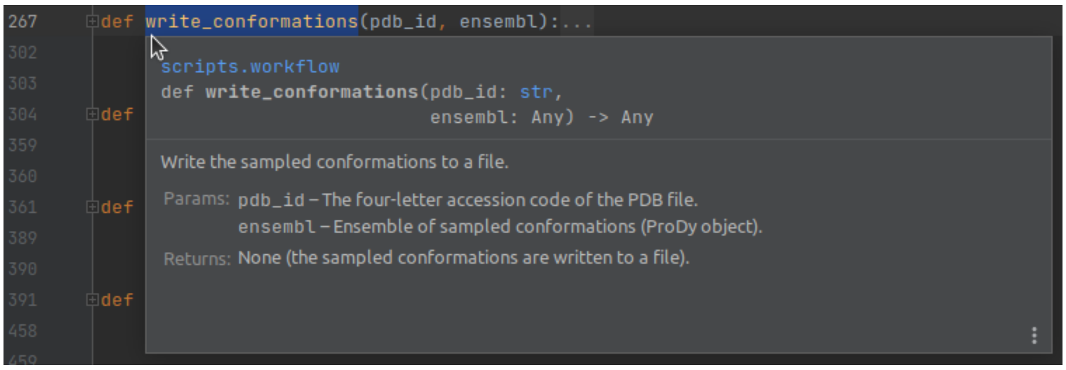
```
scripts.workflow
def sample_conformations(pdb_id: str,
                         n_confs: int = 1000,
                         rmsd: float = 1.0) -> Ensemble
```

Sample conformations from along ANM modes. The sampled conformations are saved to a file.

Params: `pdb_id` – The four-letter accession code of the PDB file.
`n_confs` – Number of conformations to sample. Default: 1000.
`rmsd` – average RMSD that the conformations will have with respect to the initial conformation (default: 1.0 Å)

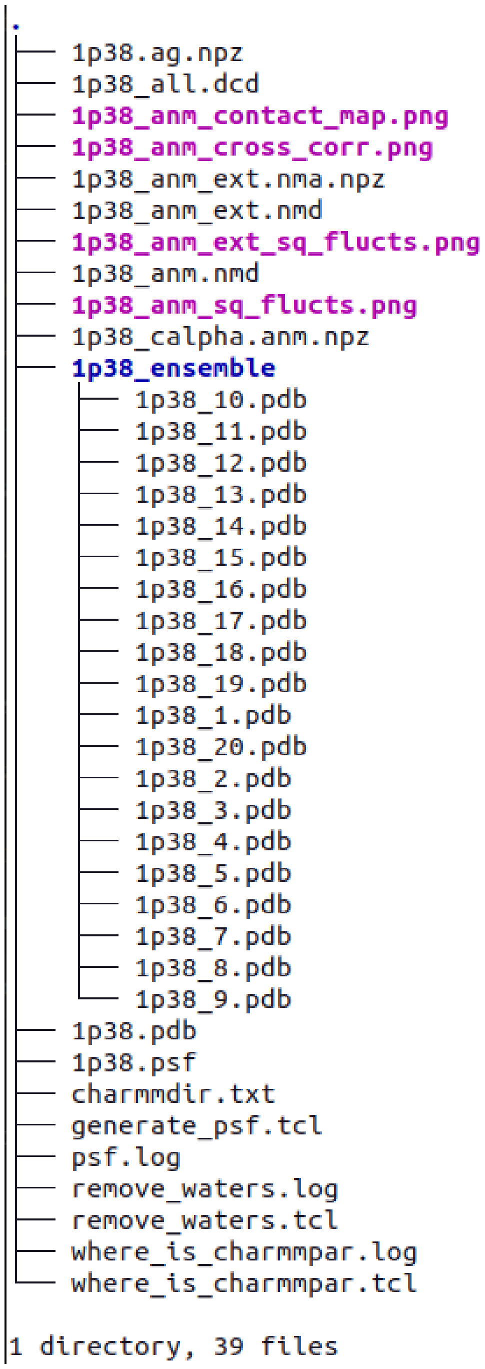
Returns: Ensemble of sampled conformations (ProDy object).

Figure 3.13: Python docstring documentation for the *sample_conformations()* method.



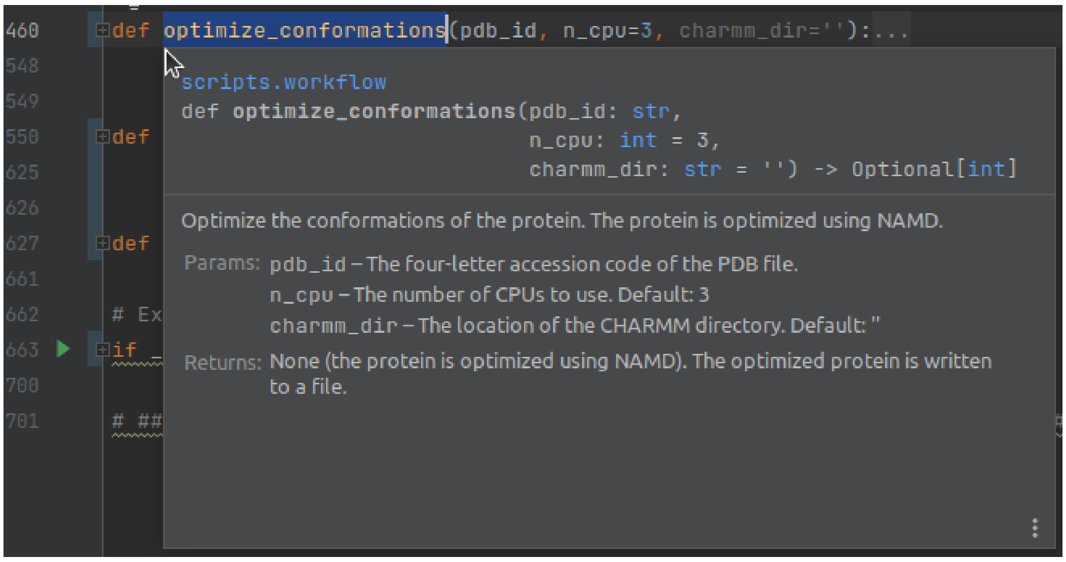
```
def write_conformations(pdb_id, ensembl):...

scripts.workflow
def write_conformations(pdb_id: str,
                        ensembl: Any) -> Any

Write the sampled conformations to a file.
Params: pdb_id – The four-letter accession code of the PDB file.
        ensembl – Ensemble of sampled conformations (ProDy object).
Returns: None (the sampled conformations are written to a file).
```

Figure 3.14: Python docstring documentation for the *write_conformations()* method.



```
.
├── 1p38.ag.npz
├── 1p38_all.dcd
├── 1p38_anm_contact_map.png
├── 1p38_anm_cross_corr.png
├── 1p38_anm_ext.nma.npz
├── 1p38_anm_ext.nmd
├── 1p38_anm_ext_sq_flucts.png
├── 1p38_anm.nmd
├── 1p38_anm_sq_flucts.png
├── 1p38_calpha.anm.npz
├── 1p38_ensemble
│   ├── 1p38_10.pdb
│   ├── 1p38_11.pdb
│   ├── 1p38_12.pdb
│   ├── 1p38_13.pdb
│   ├── 1p38_14.pdb
│   ├── 1p38_15.pdb
│   ├── 1p38_16.pdb
│   ├── 1p38_17.pdb
│   ├── 1p38_18.pdb
│   ├── 1p38_19.pdb
│   ├── 1p38_1.pdb
│   ├── 1p38_20.pdb
│   ├── 1p38_2.pdb
│   ├── 1p38_3.pdb
│   ├── 1p38_4.pdb
│   ├── 1p38_5.pdb
│   ├── 1p38_6.pdb
│   ├── 1p38_7.pdb
│   ├── 1p38_8.pdb
│   └── 1p38_9.pdb
├── 1p38.pdb
├── 1p38.psf
├── charmmdir.txt
├── generate_psf.tcl
├── psf.log
├── remove_waters.log
├── remove_waters.tcl
├── where_is_charmmpar.log
└── where_is_charmmpar.tcl

1 directory, 39 files
```

Figure 3.15: This is the tree view of the working directory with the ensemble of 20 conformations for the protein (PDB-ID: 1P38).



```
def optimize_conformations(pdb_id, n_cpu=3, charmm_dir=''):...


def


def

# Ex
if _

# ##
```

```
scripts.workflow
def optimize_conformations(pdb_id: str,
                           n_cpu: int = 3,
                           charmm_dir: str = '') -> Optional[int]
```

Optimize the conformations of the protein. The protein is optimized using NAMD.

Params: `pdb_id` – The four-letter accession code of the PDB file.
`n_cpu` – The number of CPUs to use. Default: 3
`charmm_dir` – The location of the CHARMM directory. Default: ''

Returns: None (the protein is optimized using NAMD). The optimized protein is written to a file.

Figure 3.16: Python docstring documentation for the *optimize_conformations()* method.

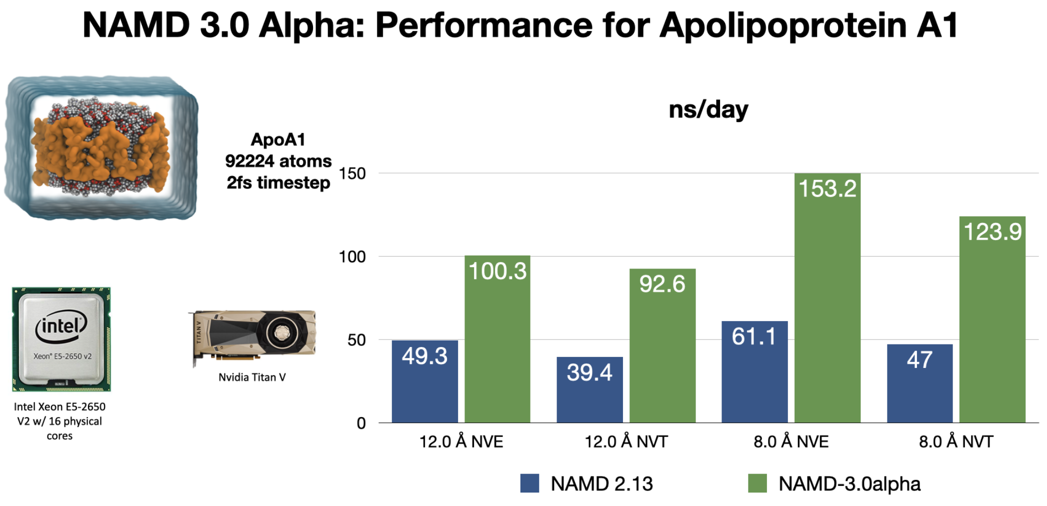
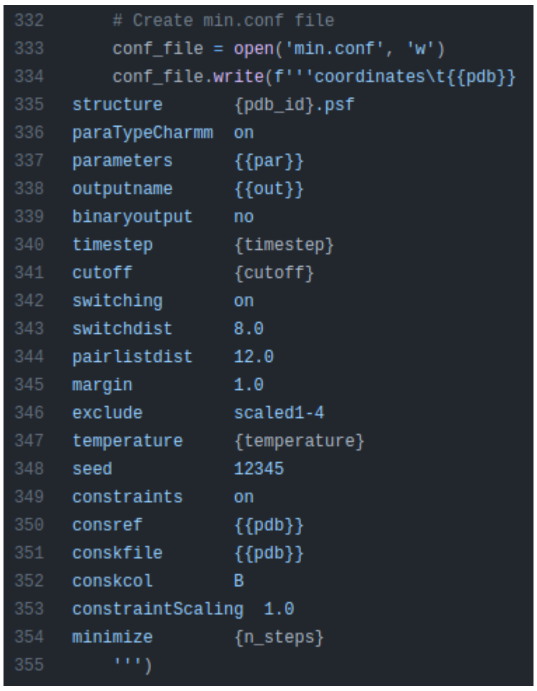
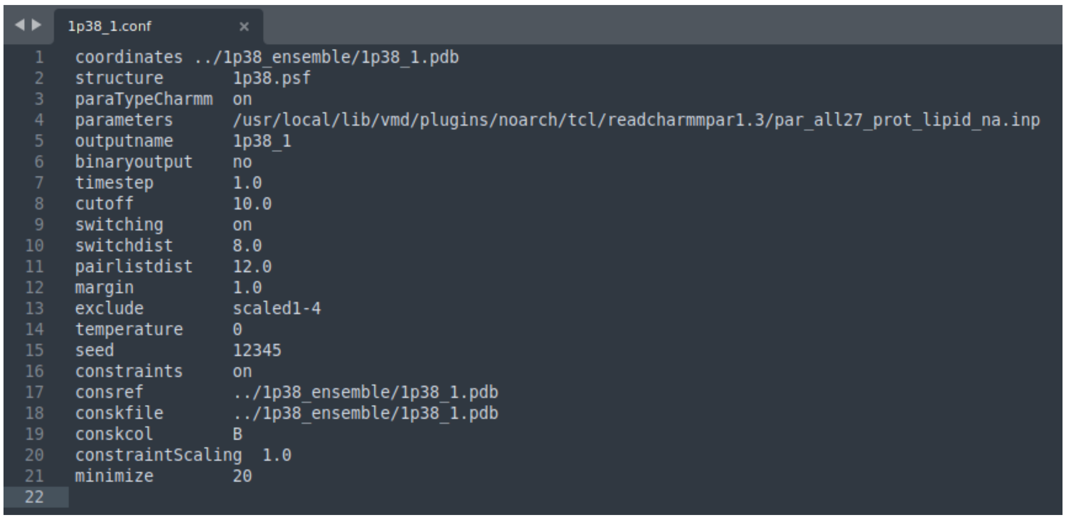


Figure 3.17: The new NAMD 3.0 GPU-resident single-node-per-replicate GPU acceleration feature provides up to 2x the performance of prior versions for appropriate MD simulations on modern GPUs. [20]



```
    # Create min.conf file
    conf_file = open('min.conf', 'w')
    conf_file.write(f'''coordinates\t{{pdb}}
structure       {pdb_id}.psf
paraTypeCharmm  on
parameters      {{par}}
outputname      {{out}}
binaryoutput    no
timestep        {timestep}
cutoff          {cutoff}
switching       on
switchdist      8.0
pairlistdist    12.0
margin          1.0
exclude         scaled1-4
temperature     {temperature}
seed            12345
constraints     on
consref         {{pdb}}
conskfile       {{pdb}}
conskcol        B
constraintScaling  1.0
minimize        {n_steps}
    ''')
```

Figure 3.18: The template for NAMD configuration files.



```
coordinates ../1p38_ensemble/1p38_1.pdb
structure       1p38.psf
paraTypeCharmm  on
parameters      /usr/local/lib/vmd/plugins/noarch/tcl/readcharmmpar1.3/par_all27_prot_lipid_na.inp
outputname      1p38_1
binaryoutput    no
timestep        1.0
cutoff          10.0
switching       on
switchdist      8.0
pairlistdist    12.0
margin          1.0
exclude         scaled1-4
temperature     0
seed            12345
constraints     on
consref         ../1p38_ensemble/1p38_1.pdb
conskfile       ../1p38_ensemble/1p38_1.pdb
conskcol        B
constraintScaling  1.0
minimize        20
```

Figure 3.19: The content of NAMD configuration file. (PDB-ID: 1P38, conformation number 1).

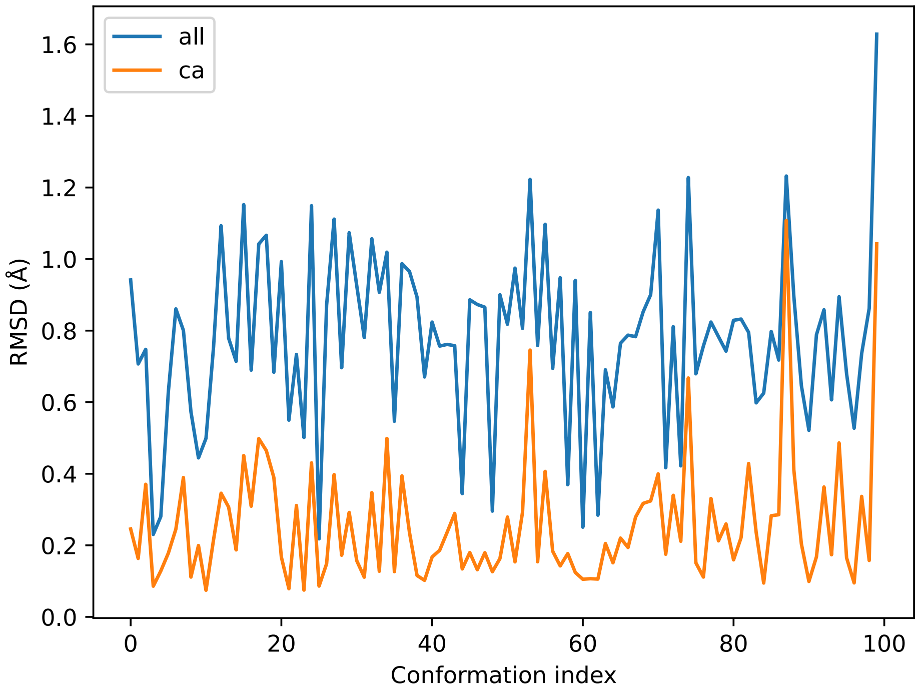


Figure 3.20: The RMSD change after refinement using NAMD (PDB-ID: 1P38).

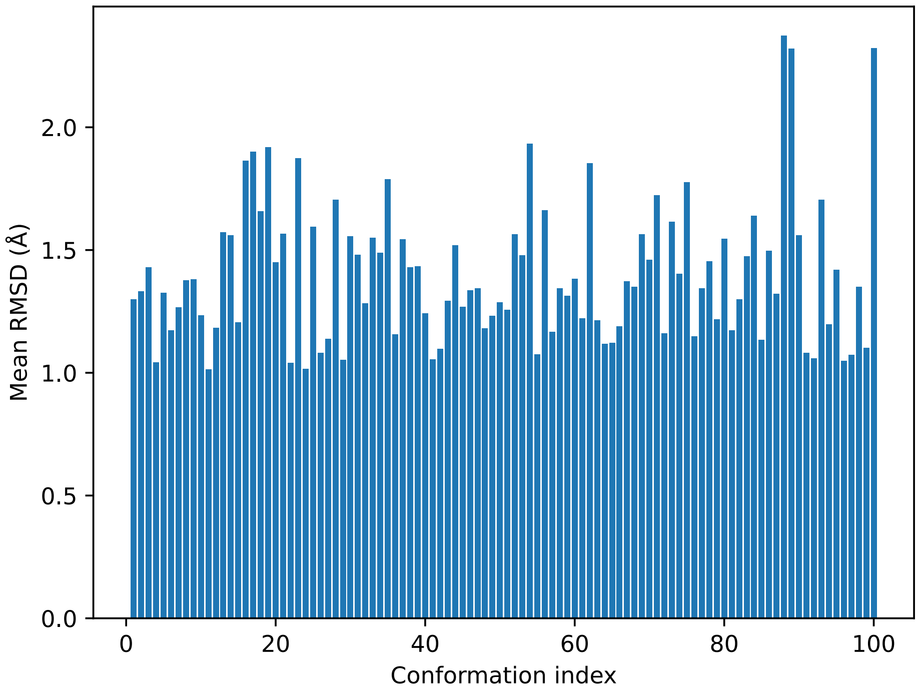


Figure 3.21: The bar plot showing the mean RMSD relative to all other conformations (PDB-ID: 1P38).

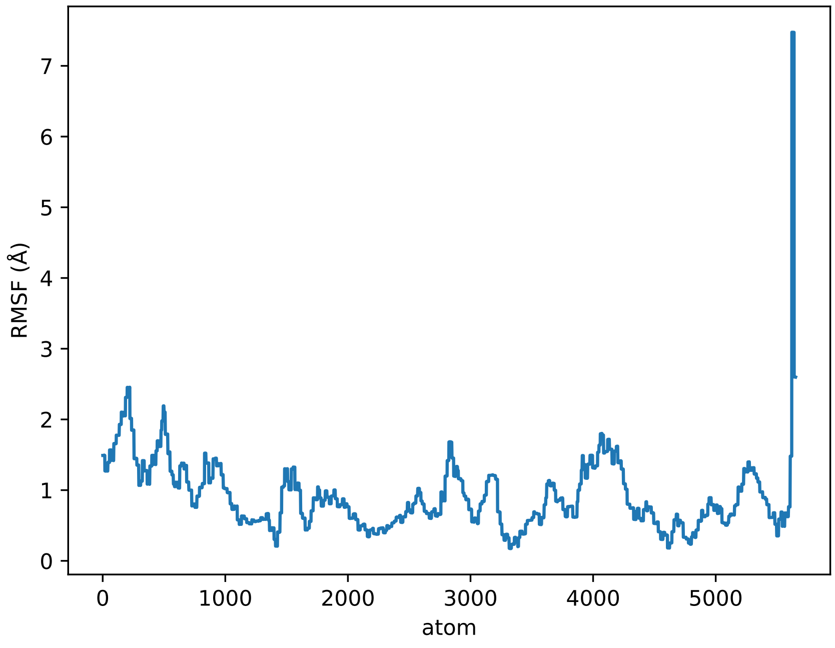


Figure 3.22: The bar plot showing the RMSF (PDB-ID: 1P38).

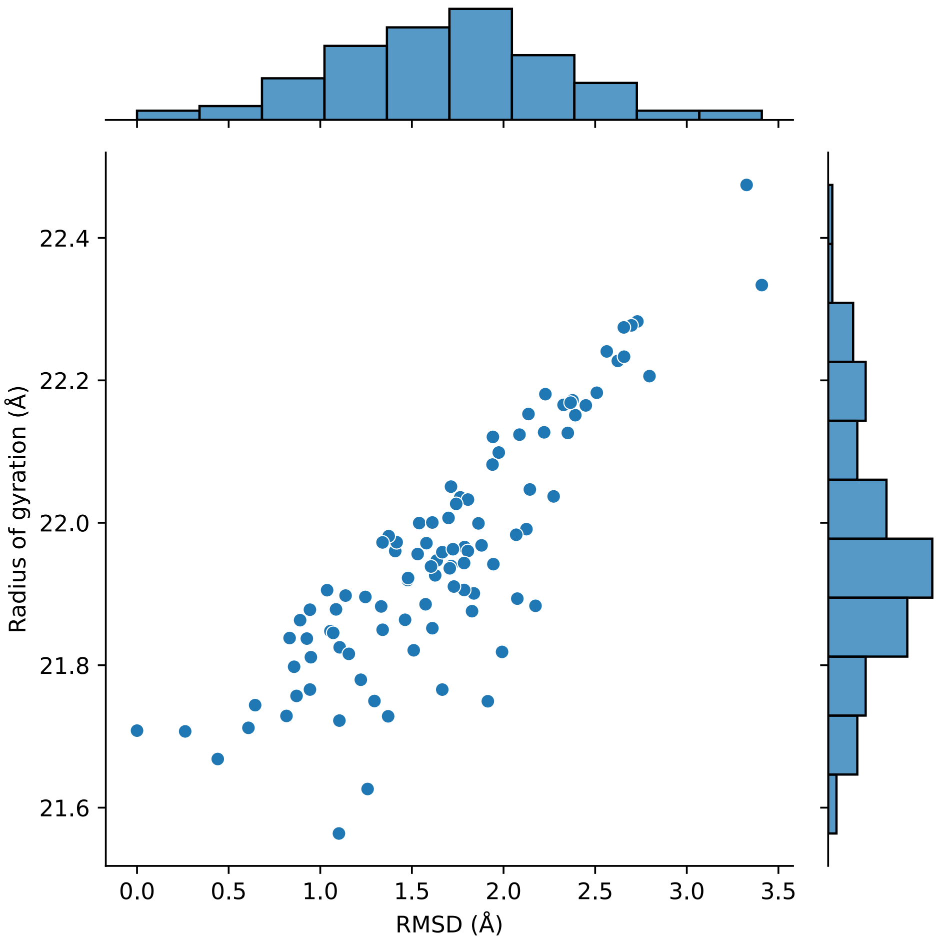
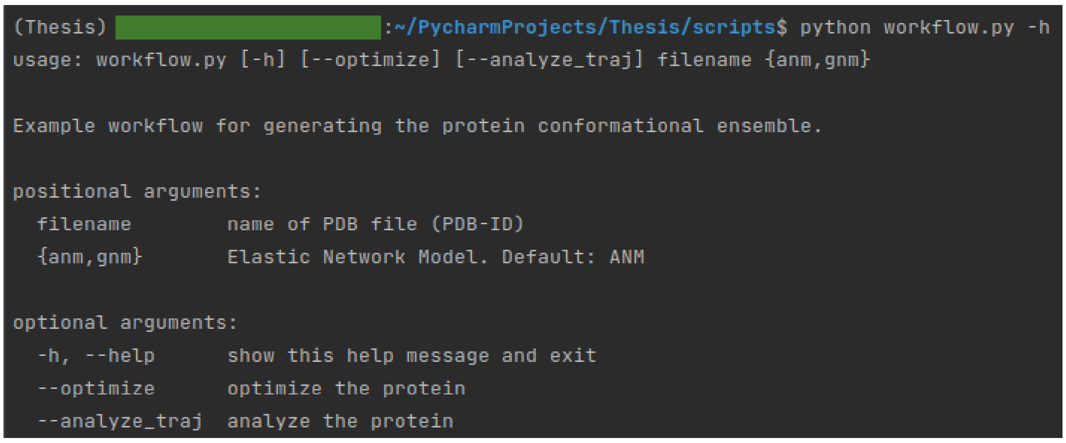


Figure 3.23: The plot showing the correlation of RMSD and radius of gyration (PDB-ID: 1P38).

## 3.3 Example workflow

The example workflow consists of every step described in the 3.2 section of this chapter and is easily available by executing the **workflow.py** script. (Figure 3.24) All stages should be performed in order of appearance.



```
(Thesis) [redacted]:~/PycharmProjects/Thesis/scripts$ python workflow.py -h
usage: workflow.py [-h] [--optimize] [--analyze_traj] filename {anm,gnm}

Example workflow for generating the protein conformational ensemble.

positional arguments:
  filename        name of PDB file (PDB-ID)
  {anm,gnm}       Elastic Network Model. Default: ANM

optional arguments:
  -h, --help      show this help message and exit
  --optimize      optimize the protein
  --analyze_traj  analyze the protein
```

Figure 3.24: The usage of workflow.py script. It requires two positional arguments to use the example workflow.

To summarize: The PDB file is parsed and prepaired using *load_pdb()*. The protein structure is shown to demonstrate that the protein was loaded successfully. Next, the method *calc_modes()* is used to calculate the normal modes using the ENM chosen from the command line. Users can define the number of modes to calculate.

The model is then extended using *extend_model()* to cover the entire structure. User is then prompted if they wish to view the model in VMD.

For ANM, the conformational ensemble is created using *sample_conformations()*. The number of conformations and threshold RMSD can be adjusted by the user to fit their needs.

After successful sampling, the *write_conformations()* method is used to save the ensemble into the new folder with the name followed by *_ensemble*.

Additionally, the ensemble can be optimized using NAMD. NAMD requires configuration files which are found and generated using *make_namd_conf()*. The geometries of the ensemble are optimized using *optimize_conformations()* method. Optimized conformations are saved into the new folder with the name followed by *_optimize*.

The last stage of generating the diverse protein ensemble is the analysis. It is performed using *analyze_conformations()*. The graph representing RMSD change (Figure 3.20) is created to help visualize the diversity of generated ensemble. The other graph represents the average RMSD relative to all other conformations (Figure 3.21). The indices of conformations that are above the set threshold are returned on the screen. Selected conformations are then copied to the new directory with the name followed by *_selected*.

If the *–analyze_traj* argument is used in example workflow, the additional analysis is performed using *analyze_pytraj()*. In this process, more detailed information about the ensemble is presented (Figures 3.22 and 3.23) This step is optional but helps users better understand generated conformations.

## 3.4 Workflow testing

Developed workflow was tested on a set of enzymes provided by *Professor Jan Brezovsky and his team.* These are enzymes catalyzing 5 distinct reaction types based on EC classification[21] covering 4 main structural classes according to SCOP[22](Tables: 3.1, 3.2, 3.3, 3.4 and 3.5):

- all alpha - Proteins containing predominantly alpha-helices
- all beta - Proteins containing predominantly beta-strands
- alpha + beta - Proteins with segregated alpha-helices and beta-strands
- alpha / beta - Proteins with alternating alpha-helices and beta-strands

Some tests did not complete due to errors during runtime.
Tables represent the tested PDB-ID and the outcome of each test:

- *ok* - indicates that the workflow was finished successfully
- *PSF error* - indicates that the error occured during generation of PSF file. This type of error need further investigation by the User and is generated by VMD. More information can be found in *psf.log* file. It can be avoided if the User uses existing PSF file - in such case the new PSF file is not generated.

Errors such as *unknown residue type XXX* are present due to decision to keep important decisions to the User. Such errors could be prevented by using aliases on problematic residue types but this approach can later yield unwanted results.

Table 3.1: Tested group: EC1

| Class | PDB | Outcome |
|---|---|---|
| all alpha | 1FJB | OK |
| | 2E39 | OK |
| | 2ZWT | OK |
| all beta | 1GP4 | PSF error (unknown residue type MSE) |
| | 1UMK | OK |
| | 4BB3 | OK |
| alpha+beta | 1FVA | PSF error (duplicate residue keys) |
| | 1MBB | OK |
| | 3TK2 | OK |
| alpha/beta | 2IQ0 | PSF error (unknown residue type 6NA) |
| | 3BUR | PSF error (duplicate residue keys) |
| | 4H97 | PSF error (duplicate residue keys) |

The developed workflow is able to provide diverse conformational ensembles for all classes of protein structures. Their biological significance and correctness can be evaluated by the Users based on provided analyses3.2.8. Few errors occured during testing. These errors are related to NAMD software and can be resolved by the Users using log files.

Table 3.2: Tested group: EC2

| Class | PDB | Outcome |
|---|---|---|
| all alpha | 1M15 | OK |
| | 1K3Y | PSF error (duplicate residue keys) |
| | 1EFY | OK |
| all beta | 1OYG | OK |
| | 4JLG | PSF error (duplicate residue keys) |
| | 1GPR | OK |
| alpha+beta | 1Q0N | OK |
| | 2PQ9 | OK |
| | 1CJW | OK |
| alpha/beta | 1Q20 | OK |
| | 1JG1 | PSF error (unknown residue type SAH) |
| | 1B8N | OK |

Table 3.3: Tested group: EC3

| Class | PDB | Outcome |
|---|---|---|
| all alpha | 5G57 | PSF error (duplicate residue keys) |
| | 1KG2 | OK |
| | 2OUN | PSF error (duplicate residue keys) |
| all beta | 1H6L | OK |
| | 1DIM | OK |
| | 1E1A | OK |
| alpha+beta | 1AKO | OK |
| | 2GG2 | OK |
| | 1BSJ | OK |
| alpha/beta | 3MVI | PSF error (duplicate residue keys) |
| | 1CVL | OK |
| | 1BF6 | PSF error (duplicate residue keys) |

Table 3.4: Tested group: EC4

| Class | PDB | Outcome |
|---|---|---|
| all alpha | 3KB9 | OK |
| all beta | 1BIC | OK |
| alpha/beta | 1A53 | OK |
| | 1AK1 | OK |

Table 3.5: Tested group: EC5

| Class | PDB | Outcome |
|---|---|---|
| alpha+beta | 1EYQ | PSF error (duplicate residue keys) |
| | 1GQZ | OK |

# Chapter 4

# Conclusion

The aim of this thesis was to create the interface to tools for approximate dynamics for use in generating and validating diverse protein ensembles. The Python package was successfully designed and developed with accompanying documentation. The aforementioned tool was designed to be used in other scripts but can be run separately from the command line to follow the example workflow. Users can use the PDB file from the local directory or automatically download one from the RCSB database. After providing PDB code users may choose between different Elastic Network Models to calculate and use them to create a diverse conformational ensemble for chosen protein which if needed can be further optimized using NAMD software.
A created ensemble can be then evaluated using provided metrics to which users can easily plot graphs.
At any step of this workflow, users can modify the parameters to personalize the output and adjust it to their needs. This enables users to select optimal settings for the generation of the ensemble, balancing the exploration of new conformations against the maintenance of the structural features of the native (input) protein.

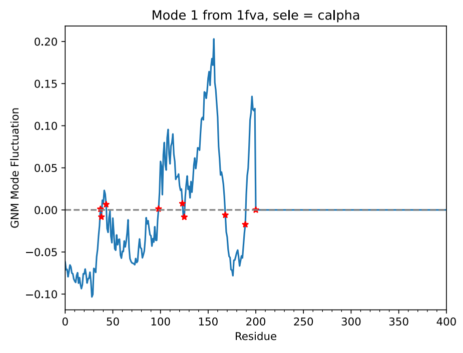
To summarize, this thesis presents a developed and documented tool. It is publicly available on Github[23] and can be used by other researchers to generate diverse protein ensembles and check their viability. Such ensembles may not be biologically viable and they should be evaluated by the Users using provided metrics. This tool can be further expanded by adding new analyses to better benefit the user experience.

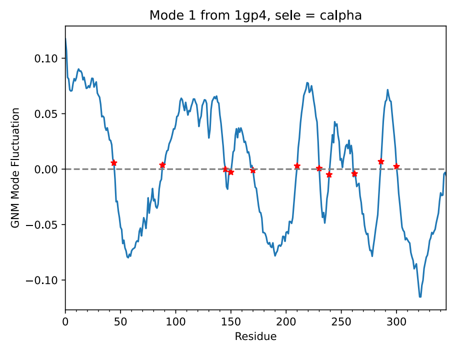
# Appendix A

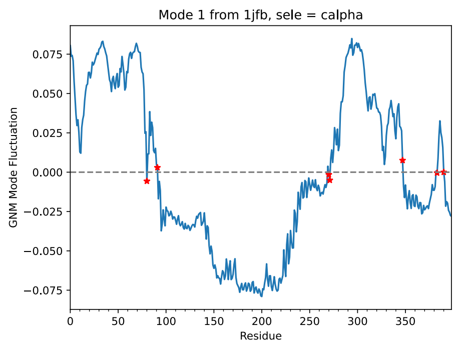
# Workflow test results

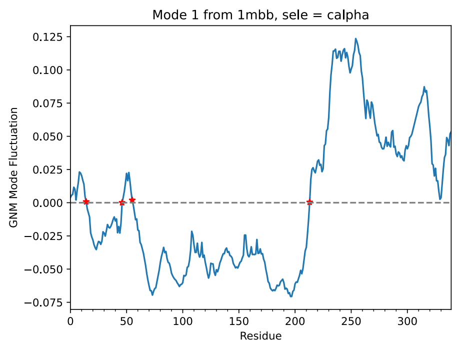
Presented below are results of GNM sampling for 5 EC groups. These tests were carried out in order to determine the capabilities of this workflow. The workflow works as intended with all 4 structural classes[22]. It proves that this workflow is capable of generating diverse conformational ensembles.

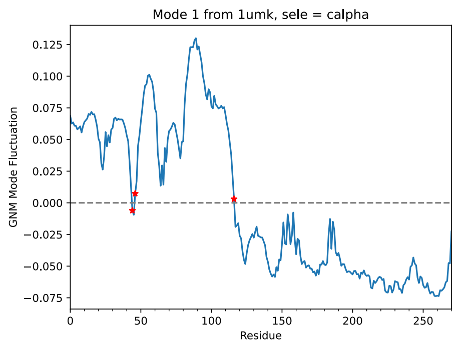
## A.1 EC1

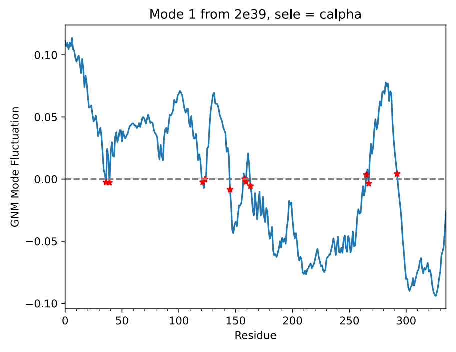

Mode 1 from 1fva, sele = calpha
GNM Mode Fluctuation
Residue

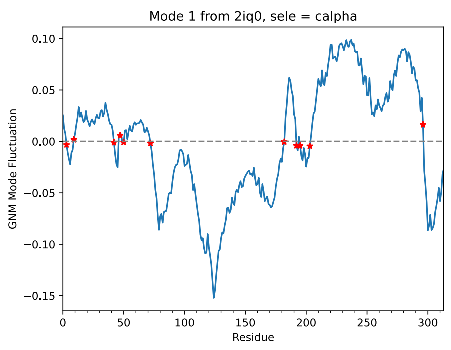

Mode 1 from 1gp4, sele = calpha
GNM Mode Fluctuation
Residue

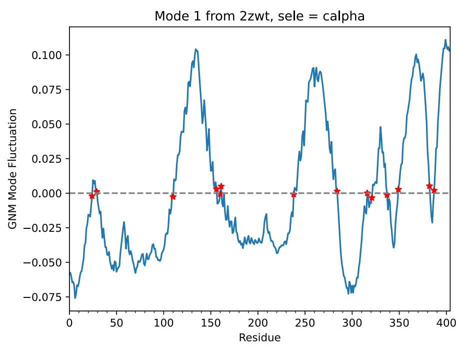

Mode 1 from 1jfb, sele = calpha
GNM Mode Fluctuation
Residue

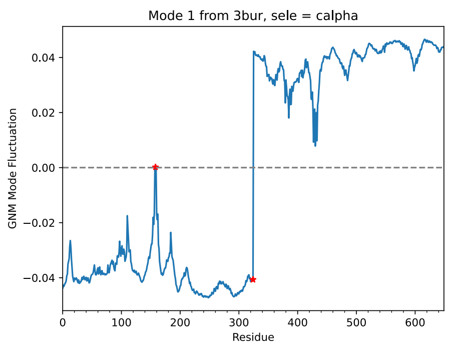

Mode 1 from 1mbb, sele = calpha
GNM Mode Fluctuation
Residue

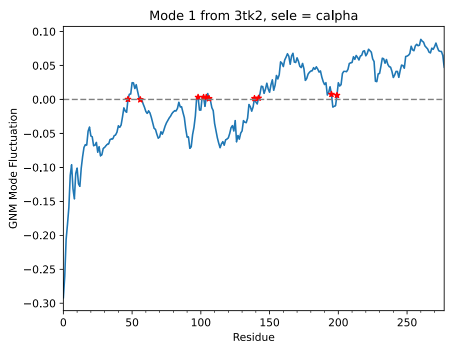

Mode 1 from 1umk, sele = calpha
GNM Mode Fluctuation
Residue

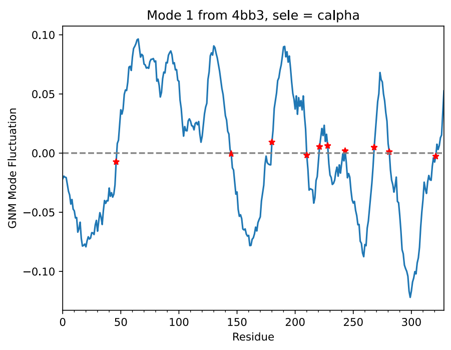

Mode 1 from 2e39, sele = calpha
GNM Mode Fluctuation
Residue

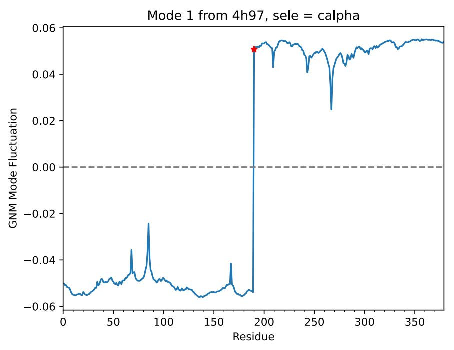

Mode 1 from 2iq0, sele = calpha
GNM Mode Fluctuation
Residue


Mode 1 from 2zwt, sele = calpha
GNM Mode Fluctuation
Residue

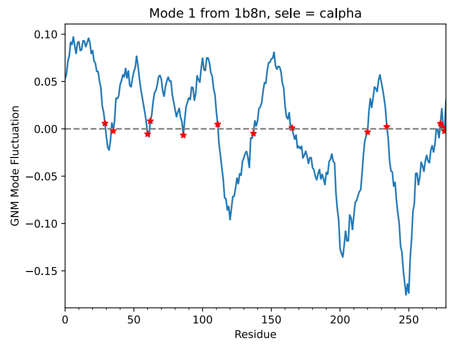

Mode 1 from 3bur, sele = calpha
GNM Mode Fluctuation
Residue

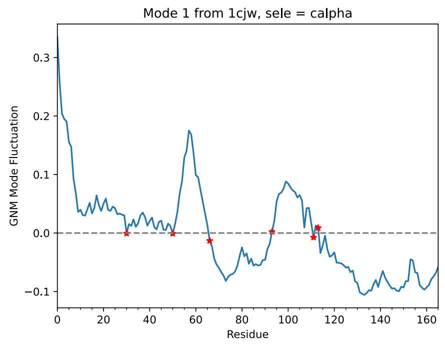

Mode 1 from 3tk2, sele = calpha
GNM Mode Fluctuation
Residue

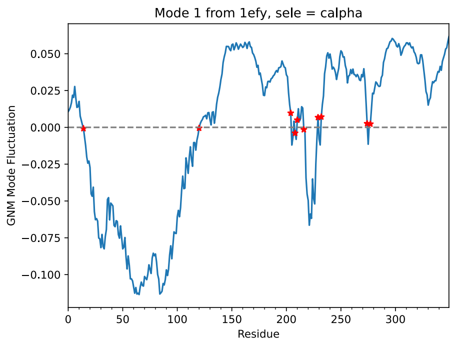

Mode 1 from 4bb3, sele = calpha
GNM Mode Fluctuation
Residue

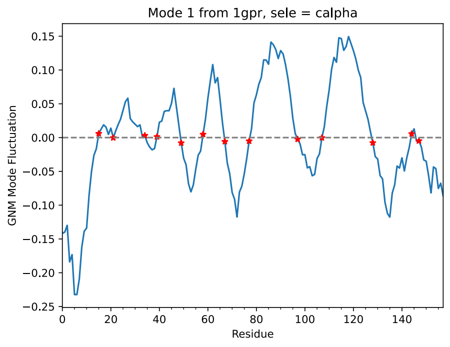

Mode 1 from 4h97, sele = calpha
GNM Mode Fluctuation
Residue

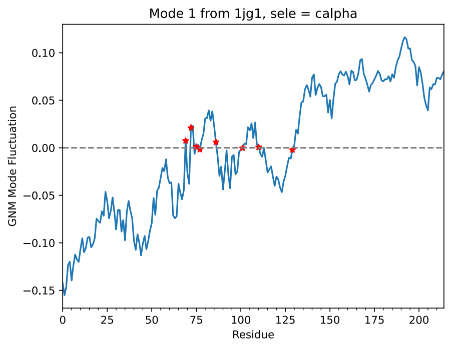
## A.2 EC2

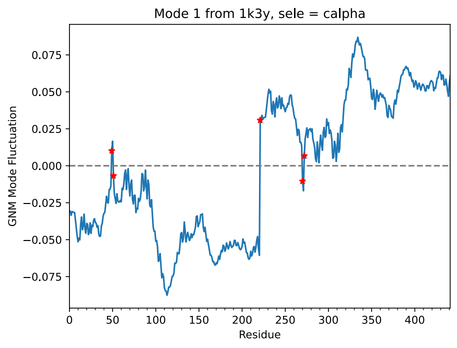

Mode 1 from 1b8n, sele = calpha
GNM Mode Fluctuation
Residue

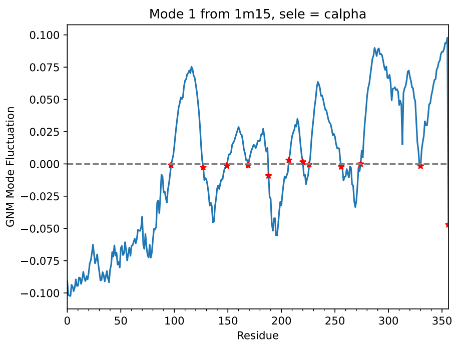

Mode 1 from 1cjw, sele = calpha
GNM Mode Fluctuation
Residue

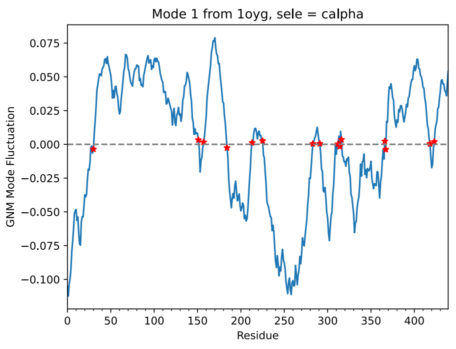

Mode 1 from 1efy, sele = calpha
GNM Mode Fluctuation
Residue

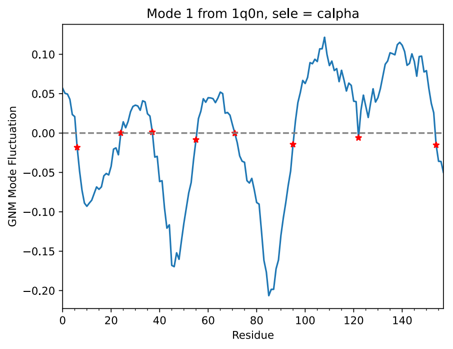

Mode 1 from 1gpr, sele = calpha
GNM Mode Fluctuation
Residue

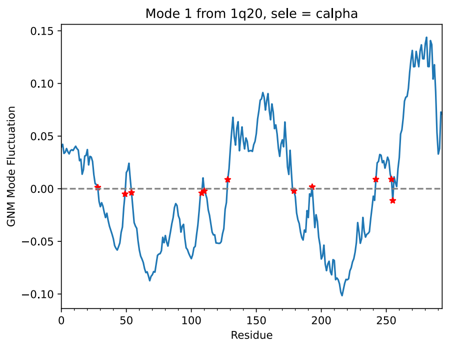

Mode 1 from 1jg1, sele = calpha
GNM Mode Fluctuation
Residue

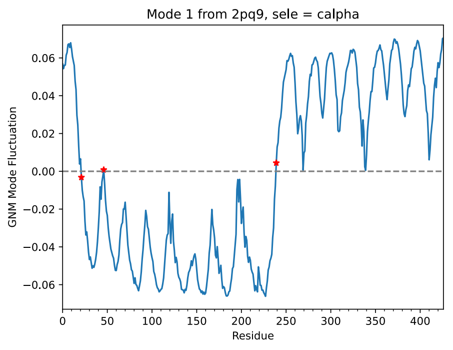

Mode 1 from 1k3y, sele = calpha
GNM Mode Fluctuation
Residue

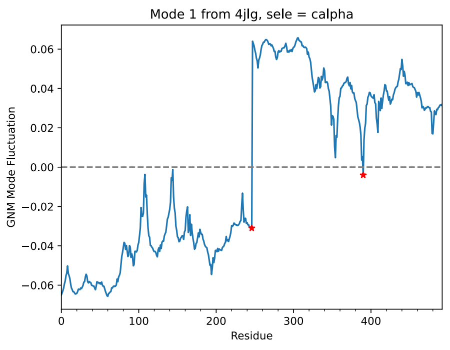

Mode 1 from 1m15, sele = calpha
GNM Mode Fluctuation
Residue


Mode 1 from 1oyg, sele = calpha
GNM Mode Fluctuation
Residue

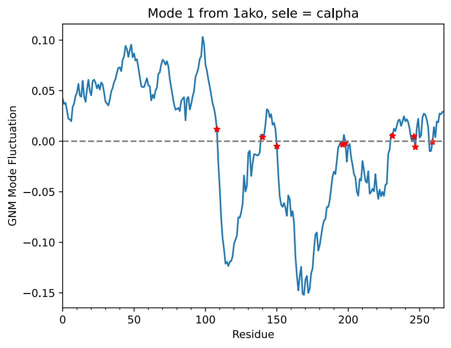

Mode 1 from 1q0n, sele = calpha
GNM Mode Fluctuation
Residue

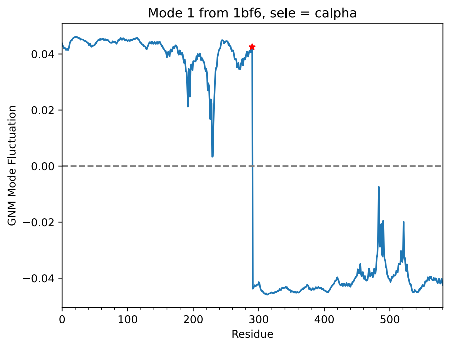

Mode 1 from 1q20, sele = calpha
GNM Mode Fluctuation
Residue

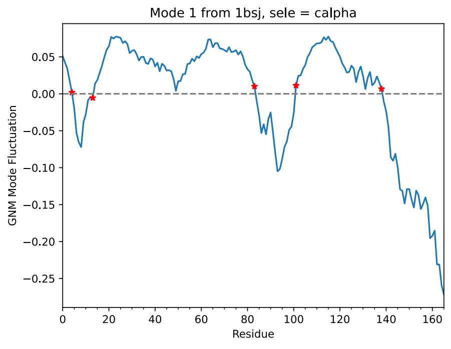

Mode 1 from 2pq9, sele = calpha
GNM Mode Fluctuation
Residue

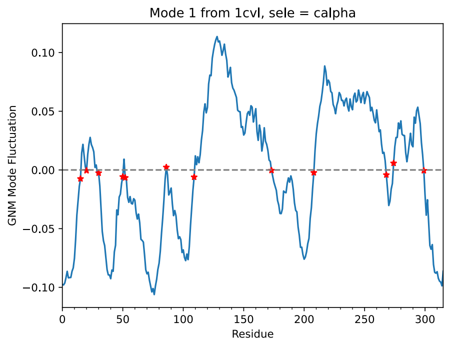

Mode 1 from 4jlg, sele = calpha
GNM Mode Fluctuation
Residue

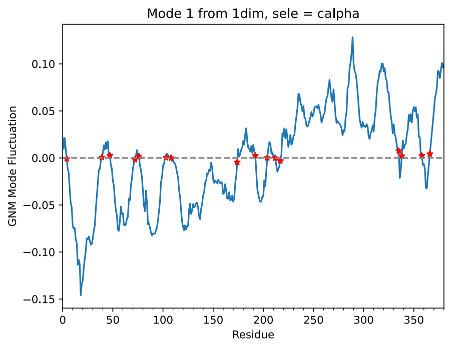
## A.3 EC3

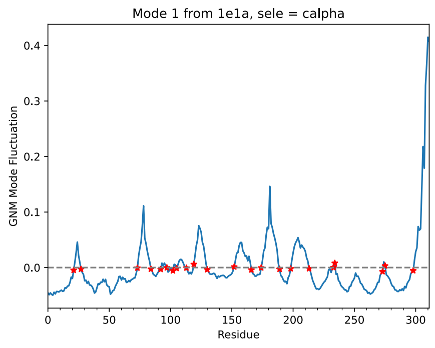

Mode 1 from 1ako, sele = calpha
GNM Mode Fluctuation
Residue

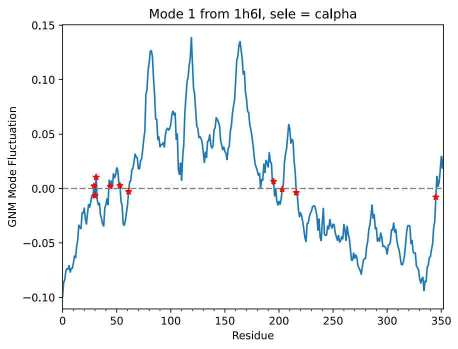

Mode 1 from 1bf6, sele = calpha
GNM Mode Fluctuation
Residue

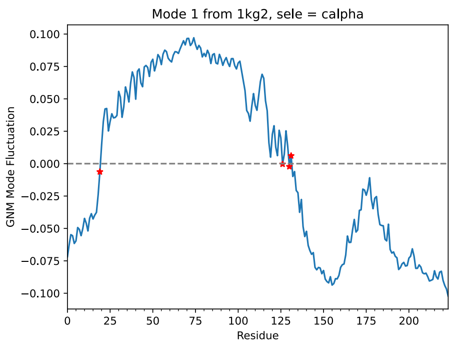

Mode 1 from 1bsj, sele = calpha
GNM Mode Fluctuation
Residue

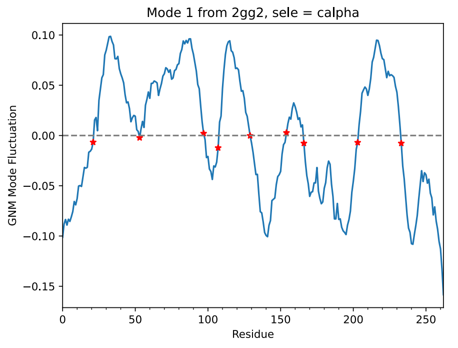

Mode 1 from 1cvl, sele = calpha
GNM Mode Fluctuation
Residue

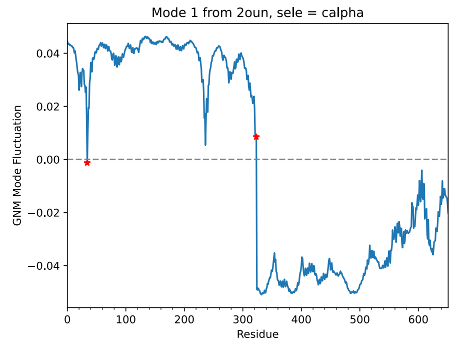

Mode 1 from 1dim, sele = calpha
GNM Mode Fluctuation
Residue

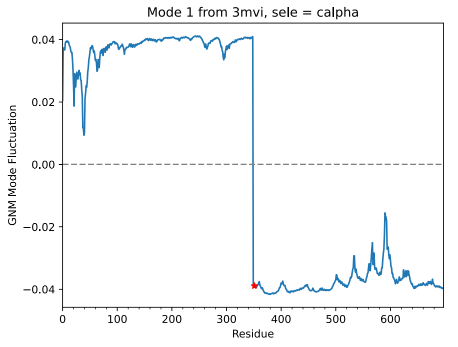

Mode 1 from 1e1a, sele = calpha
GNM Mode Fluctuation
Residue

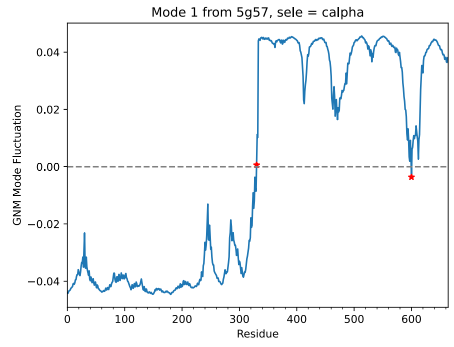

Mode 1 from 1h6l, sele = calpha
GNM Mode Fluctuation
Residue


Mode 1 from 1kg2, sele = calpha
GNM Mode Fluctuation
Residue

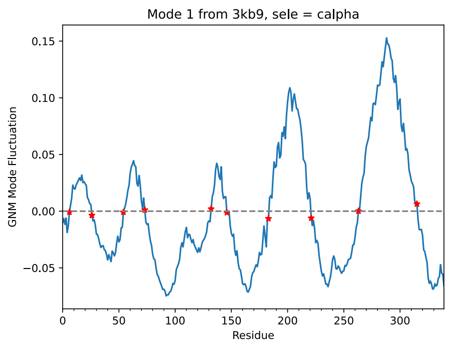

Mode 1 from 2gg2, sele = calpha
GNM Mode Fluctuation
Residue

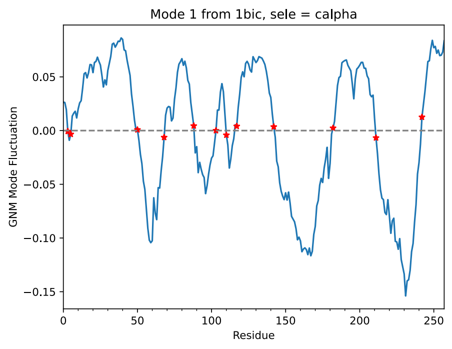

Mode 1 from 2oun, sele = calpha
GNM Mode Fluctuation
Residue

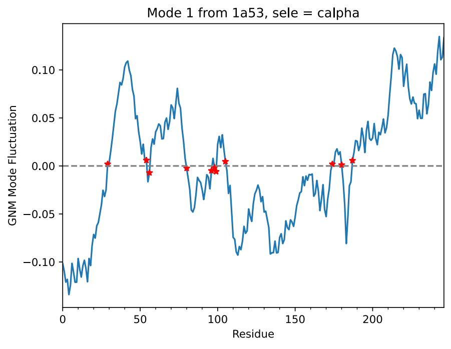

Mode 1 from 3mvi, sele = calpha
GNM Mode Fluctuation
Residue

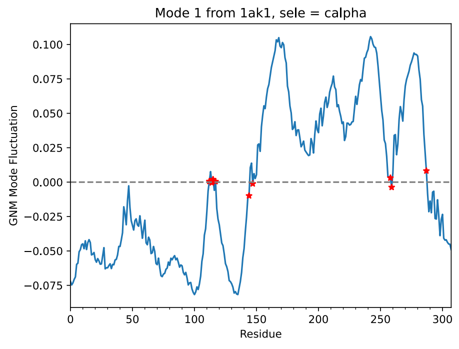

Mode 1 from 5g57, sele = calpha
GNM Mode Fluctuation
Residue

## A.4 EC4

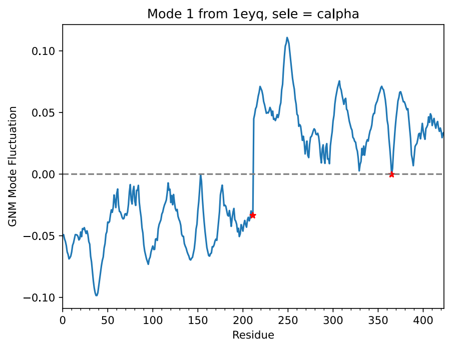

Mode 1 from 3kb9, sele = calpha
GNM Mode Fluctuation
Residue

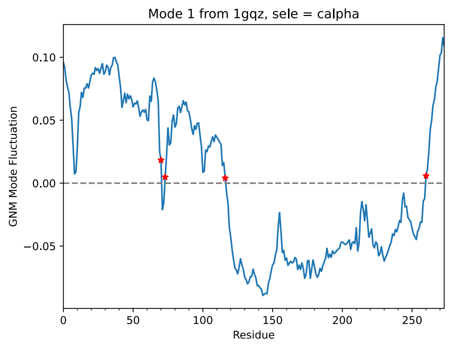

Mode 1 from 1bic, sele = calpha
GNM Mode Fluctuation
Residue


Mode 1 from 1a53, sele = calpha
GNM Mode Fluctuation
Residue


Mode 1 from 1ak1, sele = calpha
GNM Mode Fluctuation
Residue

## A.5 EC5


Mode 1 from 1eyq, sele = calpha
GNM Mode Fluctuation
Residue


Mode 1 from 1gqz, sele = calpha
GNM Mode Fluctuation
Residue

# Bibliography

[1] DE Koshland and F Haurowitz. Protein: Definition, Structure, & Classification, 2019.

[2] Bruce Alberts, Alexander Johnson, Julian Lewis, Martin Raff, Keith Roberts, and Peter Walter. The shape and structure of proteins. In *Molecular Biology of the Cell. 4th edition*. Garland Science, 2002.

[3] Nature education. Protein Structure — Learn Science at Scitable, 2010.

[4] A G-Preciado, M Peimbert, and E Merino. Genome sequence databases: types of data and bioinformatic tools. *Encyclopedia of Microbiology (Third Edition)*, 2009.

[5] David S. Goodsell. PDB101: Learn: Guide to understanding PDB data: Introduction.

[6] Yuichi Togashi and Holger Flechsig. Coarse-grained protein dynamics studies using elastic network models. *International journal of molecular sciences*, 19(12):3899, 2018.

[7] Jane R Allison, Peter Varnai, Christopher M Dobson, and Michele Vendruscolo. Determination of the free energy landscape of $\alpha$-synuclein using spin label nuclear magnetic resonance measurements. *Journal of the American Chemical Society*, 131(51):18314–18326, 2009.

[8] X Salvatella. Understanding protein dynamics using conformational ensembles. *Protein Conformational Dynamics*, pages 67–85, 2014.

[9] J Andrew McCammon, Bruce R Gelin, and Martin Karplus. Dynamics of folded proteins. *nature*, 267(5612):585–590, 1977.

[10] James C Phillips, David J Hardy, Julio DC Maia, John E Stone, João V Ribeiro, Rafael C Bernardi, Ronak Buch, Giacomo Fiorin, Jérôme Hénin, Wei Jiang, et al. Scalable molecular dynamics on CPU and GPU architectures with NAMD. *The Journal of chemical physics*, 153(4):044130, 2020.

[11] Wilfred F van Gunsteren, Jožica Dolenc, and Alan E Mark. Molecular simulation as an aid to experimentalists. *Current opinion in structural biology*, 18(2):149–153, 2008.

[12] Timothy R Lezon, Indira H Shrivastava, Zheng Yang, and Ivet Bahar. Elastic network models for biomolecular dynamics: theory and application to membrane proteins and viruses. *Handbook on biological networks*, pages 129–58, 2009.

[13] Eran Eyal, Lee-Wei Yang, and Ivet Bahar. Anisotropic network model: systematic evaluation and a new web interface. *Bioinformatics*, 22(21):2619–2627, 2006.

[14] Ivet Bahar, Ali Rana Atilgan, and Burak Erman. Direct evaluation of thermal fluctuations in proteins using a single-parameter harmonic potential. *Folding and Design*, 2(3):173–181, 1997.

[15] Roderick E Hubbard and Muhammad Kamran Haider. Hydrogen bonds in proteins: role and strength. *eLS*, 2010.

[16] Embl-Ebi. Levels of protein structure – secondary.

[17] What is python? executive summary.

[18] She Zhang, James M Krieger, Yan Zhang, Cihan Kaya, Burak Kaynak, Karolina Mikulska-Ruminska, Pemra Doruker, Hongchun Li, and Ivet Bahar. ProDy 2.0: increased scale and scope after 10 years of protein dynamics modelling with Python. *Bioinformatics*, 37(20):3657–3659, 2021.

[19] Amber-MD. Amber-MD/pytraj: Python interface of cpptraj.

[20] NAMD 3.0 Alpha, GPU-resident single-node-per-replicate test builds.

[21] Enzyme Nomenclature. Recommendations of the nomenclature committee of the international union of biochemistry and molecular biology on the nomenclature and classification of enzymes, 1992.

[22] Antonina Andreeva, Eugene Kulesha, Julian Gough, and Alexey G Murzin. The scop database in 2020: expanded classification of representative family and superfamily domains of known protein structures. *Nucleic acids research*, 48(D1):D376–D382, 2020.

[23] Mikołaj Koszczyc. mikoszczyc/Thesis: v1.0.0, 2022.

Poznan University of Technology
Faculty of Computing and Telecommunication
Institute of Computing Science

Typeset using LaTeX.