ProDy

Protein Dynamics & Sequence Analysis

Adaptive ANM

Release

James Krieger, She Zhang

April 22, 2024

CONTENTS

1 Introduction **1**
1.1 Required Programs . . . 1
1.2 Getting Started . . . 1

2 Calculations **3**
2.1 Loading the structure and running adaptive ANM . . . 3
2.2 Analysis . . . 3

Bibliography **5**

CHAPTER ONE

INTRODUCTION

This tutorial shows how to transition between two conformational states using adaptive ANM. The example used in this tutorial is the GroEL chaperonin in the R” and T states (chain A of PDB structures 1GRU and 1GR5), which we previously studied using this method (see Figure 4 of *[ZY09]*).

1.1 Required Programs

The latest version of **ProDy_** is required.

1.2 Getting Started

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

```
$ ipython
```

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

We have included these imports in every part of the tutorial, so that code copied from the online pages is complete. You do not need to repeat imports in the same Python session.

CHAPTER

TWO

CALCULATIONS

Here are the required imports again. You do not need to repeat them if you are still in the same python session.

2.1 Loading the structure and running adaptive ANM

First, we parse the structures that we want to analyse with Adaptive ANM. For this tutorial, we will use chain A of PDB structures 1GRU and 1GR5, which correspond to the R" and T states, which we will call *r* and *t*.

We load the structures with only calpha atoms, which we use in downstream steps.

In order to have the same number of atoms in both structures, we use the function `alignChains()` to create `AtomMap` objects with the same number.

As *r* has more atoms (524) than *t* (517), we provide *t* first to find the common atoms that are present in *t*:

This new implementation of Adaptive ANM increases the number of modes calculated each cycle depending on which ones were used in the previous cycles and their overlaps. We can give it keyword arguments for the maximum number of modes to use (**3N-6** by default for **N** atoms), starting number of modes (20 by default), and other ANM parameters. These parameters can also be presented to the functions below.

Next, we can run AdaptiveANM calculations simply by

The implementation in **ProDy_** also provides 3 ways of running calculations, namely, *AANM_ALTERNATING*, *AANM_ONEWAY*, and *AANM_BOTHWAYS*. *AANM_ALTERNATING* updates *r* and *tmap* simultaneously as in the original Adaptive ANM (*[ZY09]*) with minor variations, whereas *AANM_ONEWAY* and *AANM_BOTHWAYS* carry out the calculations from either just one or both directions (starting with A and going until convergence then continuing from B). This behavior is controlled by the *mode* argument:

2.2 Analysis

`calcAdaptiveANM()` creates an `Ensemble` object that has all the generated, intermediate conformations that bridges *r* and *tmap*. The reference coordinates of *ens* are set to those of *tmap*, so that we can verify the result by checking the RMSDs of the conformations:

We can also perform any other analysis that is applicable to a `Ensemble`: object. Other quantities that may be useful for debugging or validation purposes can be obtained through assigning a callback function. For example, to extract the number of modes used in each iteration, we can write the following function to access and store the value:

Note that **N_MODES** needs to be defined outside the function, at a global scope, in order to save the value for each iteration. **modes** is a `ModeSet` object that gives you the mode(s) selected for deform the structure in an iteration. You have the access to all the properties of **modes**, and therefore the whole `ANM`, but here we are only evaluating the number of selected modes using `len()`. Please check out the documentation of `calcAdaptiveANM()` for a complete list of accessible quantities.

Now, we pass the callback function to `calcAdaptiveANM()` as follows:

And check the number of modes being selected in each iteration:

../template/acknowledgments.rst

[ZY09] Yang Z, Májek P, Bahar I. Allosteric transitions of supramolecular systems explored by network models: application to chaperonin GroEL. *PLoS Comput Biol.* **2009** 5(4):e1000360.