Supplementary Material: Coarse-grained nucleic acid - protein model for hybrid nanotechnology

Jonah Procyk[a], Erik Poppleton[a], Petr Šulc[a]

[a]*School of Molecular Sciences and Center for Molecular Design and Biomimetics, The Biodesign Institute, Arizona State University, 1001 South McAllister Avenue, Tempe, Arizona 85281, USA*

Overview

This Supplement introduces the file formats and parameters required to run simulations of the ANM-oxDNA models that are available in the anm-oxDNA repository at `https://github.com/sulcgroup/anm-oxdna`. There are 6 new types of interactions and these are:

1. DNANM: Simulates DNA and Proteins in system, Uses oxDNA2 model for DNA and classic ANM for proteins
    - Available Options: Sim Types: (MC, MD), GPU Version: Yes
2. RNANM: Simulates RNA and Proteins in system, Uses oxRNA2 model for RNA and classic ANM for proteins
    - Available Options: Sim Types: (MC, MD), GPU Version: Yes
3. DNACT: Simulates DNA and Proteins in system, Uses oxDNA2 model for DNA and ANMT for proteins
    - Available Options: Sim Types: (MC, MD), GPU Version: Yes
4. RNACT: Simulates DNA and Proteins in system, Uses oxDNA2 model for RNA and ANMT for proteins
    - Available Options: Sim Types: (MC, MD), GPU Version: Yes
5. AC: Simulates Proteins alone, Uses classic ANM for proteins
    - Available Options: Sim Types: (MC, MD), GPU Version: No
6. ACT: Simulates Proteins alone, Uses ANMT for proteins
    - Available Options: Sim Types: (MC, MD), GPU Version: No

All hybrid Models (DNANM, RNANM, DNACT, or RNACT) are capable of simulating systems consisting of just proteins, just DNA/RNA, or proteins and DNA/RNA. The models currently to not support a system which features both DNA and RNA at the same time.

Preparing a Simulation

Four files are required for any of the new Interactions: the Input file, Parameter file, Topology file, and the Configuration (or dat) file. The one unique file for every simulation involving the protein model is the Parameter File. If no protein is in the simulation, simply set 'parfile=none' in the input file.

The **Recommended Method** to prepare a system for simulation is to use our scripts available in the /ANMUtils directory in `https://github.com/sulcgroup/anm-oxdna`. To see examples of our scripts in preparing simulation files make sure to check out the /examples directory in /ANMUtils. A brief overview of the scripts is provided below.

Each of the following subsections will cover the composition of each file type necessary for a simulation and their specific options.

Topology File

The general format of the standard oxDNA topology file is preserved with the addition of this model, albeit with a few caveats. The most important consideration is that the protein strands must all be declared together. They must be declared either before or after the declaration of all DNA or RNA particles. To declare a Protein strand the strand id must be negative and starts from -1 decrementing 1 per each additional protein strand. The base ids are almost the same as the standard oxDNA format, except the one letter code for each Amino Acid is supported in the base field. In the model's current implementation the amino acid identity is arbitrary as any choice has the exact same parameters.

In the general oxDNA format, the 3' and 5' neighbors are listed directly after the base field. The same holds true for proteins except the 3' neighbor corresponds to the N-terminus neighbor and the 5' neighbor corresponds to the C-terminus neighbor. Unlike the original oxDNA format, Protein neighbors aren't limited to their N-terminus and C-terminus neighbors. All connections in the Anisotropic Network are defined in the topology file.

To account for this, the index of each amino acid (with a higher index than the particle being declared) bonded to the particle being declared must be listed after the N-terminus and C-terminus neighbors. One major change from the typical oxDNA format is the header line of the file. Usually its just two numbers with the number of strands and total particles. However, for any hybrid model five numbers are needed. The first two are the familiar number of strands and total number of particles. The next three are: the number of DNA/RNA particles, the number of protein particles, and the number of DNA/RNA strands respectively. If either DNA/RNA or protein is absent from the simulation, the simulation will run regardless as long as 0's are properly placed in the header fields.

Topology File Header Examples

- Simulation containing just DNA:

```
#Example: One 14mer strand of DNA header
1 14 14 0 1
```

- Simulation containing just a protein:

```
#Example: One 104 residue protein header
1 104 0 104 0
```

- A hypothetical 2 DNA particles and 3 Protein residues hybrid Topology file:

```
2 5 2 3 1              #header
-1 V -1 1 2            #Particle 0, AA Valine bonded to neighbors 1 & 2
-1 A 0 2               #Particle 1, AA Alanine bonded to neighbors 0 & 2
-1 T 1 -1              #Particle 2, AA Threonine bonded to neighbor 1
1 A -1 5               #Particle 4, DNA Adenine bonded to particle 5
1 G 4 -1               #Particle 5, DNA Guanine bonded to particle 4
```

- Switching the Order of the DNA and Protein residues makes no difference, so the below shows the same system with a different topology format:

```
2 5 2 3 1              #header
1 A -1 1               #Particle 0, DNA Adenine bonded to particle 1
1 G 0 -1               #Particle 1, DNA Guanine bonded to particle 0
-1 V -1 3 4            #Particle 2, AA Valine bonded to neighbors 3 & 4
-1 A 2 4               #Particle 3, AA Alanine bonded to neighbors 2 & 4
-1 T 3 -1              #Particle 4, AA Threonine bonded to neighbor 3
```

Input File

There are three different options for running any hybrid simulation (CUDA, MD CPU, MC CPU). All three require the interaction type field in the input file to be set to their respective Interaction name ex. "DNANM".

The CPU backend (MD or MC) requires no Interaction specific options except for the one required key "parfile =" which must be set to the Parameter File (see next section) or 'none'. The 'none' option should only be used when there is no protein present in the system you are trying to simulate.
For CUDA hybrid Interactions, all of the requirements for the CPU backend still stand. However, additional parameters in the input file are required to use CUDA as your simulation backend. In addition not all CUDA options implemented in oxDNA are currently supported. Currently you must use a Verlet List as well as an edge based approach. The required options as written in an Input File are listed below:

```
backend = CUDA
CUDA_Sort_Every = 0
CUDA_list = verlet
use_edge = 1
edge_n_forces = 1
```

An additional option you have with the CUDA Backend is a backend precision of mixed. This is faster than backend precision double while being almost as precise. To enable this use the following:

```
backend_precision = mixed
```

Parameter File

The Parameter file lists the necessary parameters for each spring potential in the network. Specifically it identifies the indices of the particles, their equilibrium distance, and their spring constant. Using the ANMT model changes this format slightly as shown later.
No matter what, the first line of the parameter file is a single number corresponding to the number of protein particles in the system. All lines under that follow this format with each field separated by a single space:

```
1. particle_i_index
2. particle_j_index
3. potential_type
4. equilibrium_distance
5. spring_constant
```

Example Parameter File for a system of 2 Protein Particles with a connection between Protein Particles 0 and 1 looks like this:

```
2
1 2 s 1.02 5.0
```

Here are some important considerations concerning each field of the parameter file:

- The particle indexes are the same as those in the topology file. When listing a bond, index i is always less than index j. Keeping this notation ensures each bond is accounted for only once.
- The only potential type currently supported is the spring potential. The character **s** needs to be in this field.
- The equilibrium distance is the distance between the α-Carbons at index i and index j. This is calculated from the PDB coordinates and is in simulation distance units (1 unit = 0.8518 nm). It must be in float format and non-negative.
- The spring constant will be the same for every bond in the network in the classic ANM. Must be in the form of a float. Using the HANM or mANM will have unique spring constants. The spring constants calculation is provided in our scripts. The spring constant value is in simulation units (1 unit = 57.09 pN/nm).

The ANMT model has additional fields on the parameter file. In addition to the 5 shown above, ONLY backbone connections (denoted by j_index - i_index = 1) have the following fields:

```
6. 4 Equilibrium Angle Constraints (Floats ranging from -1 to 1)
7. 2 Force Constants for the Bending and Torsional Modulation  #OPTIONAL
```

A couple of notes concerning the additional fields:

- The Equilibrium Angle Constraints are four floats ranging from -1 to 1 separated by a space between each one. Make sure the correct interaction is being used as an ANM Interaction (DNANM, RNANM) cannot read the parameter file of an ANMT Interaction (DNACT, RNACT)
- The force constants for the bending and torsional modulation can be declared in two ways. 1) Globally (for all protein backbone connections) - This is set by using the keys 'bending_k =' and 'torsion_k =' in the input file. 2) Per backbone connection - This is set by adding two numbers to the end of all backbone connections in the parameter file. By default all ANMT Interactions will assume a global declaration of the force constants if the aforementioned keys are present in the Input file. The absence of the keys indicates to the program that the force constants should be present in the parameter file. Either way, the logfile will explicitly tell you where the force constants were found. The force constants are referred to as kb and kt in the logfile.

Configuration File

Protein Particles have all the same fields as their DNA counterparts. For a detailed description of the configuration file please see `https://dna.physics.ox.ac.uk/index.php/Documentation#Configuration_file`

Scripts Overview

The scripts contains a library of python classes and functions to help convert from PDB files into our Model. It also includes a Modeller implementation for the modelling of missing residues in PDB structures.

The scripts contain 5 classes for building specific structures:

1. ANM - The classic ANM model, the other classes derive from here
2. ANMT - ANM with bending/torisional modulation
3. HANM - Fits B-factors to experimental observed
4. mANM - Multiscale ANM
5. peptide - ANMT implmentation of a Peptide

Below is an example script to demonstrate the Python module's intended usage:

```
import models as m

pdbfile = '/Desktop/tmp/myprotein.pdb'

# The first function call should always be to read in information
# on the PDB structure via the function get\_pdb\_info()

experimental_bfactors, xyz_coordinates = get_pdb_info(pdbfile)

# The optional parameter returntype in get_pdb_info can be altered to return information
# including sequence, chainids, rigid body side chain vectors and other pdb info
# See the examples in the anm-oxDNA /ANMUtils directory for detailed usage

# Now we initialize a model using just the coordinates and B factors

our_anm = m.ANM(xyz_coordinates, experimental_bfactors, T=300, cutoff=13)

# T is temp in Kelvin at which the protein B factors were determine
# Cutoff is the rmax value in Angstroms
# As a guideline, i usually start with 12-13 A and will go up
# as high as 18A

# !Cutoff values must be set upon initialization, however creating
# several models at different rmaxs and comparing is easy!
```

```
# The ANM class serves as a base for the other classes
# which all others (except peptide) inherit from

# In all classes (except peptide) there is a one-shot function that
# automatically evaluates the analytical B-factors for that model.

#For ANM

our_anm.calc_ANM_unitary()

# The above function does the following:
# 1) Evaluates B-factors via SVD of Hessian
# 2) Automatically fits the spring constant to best fit with Experimental B-factors
# 3)Optional 'cuda' parameter is recommended for large structures (See /ANMUtils)

# Using Matplotlib we can plot our calculated B factors vs. the Experimental B factors

our_anm.anm_compare_bfactors(bfactor_comparison_image)

# saves figure of compared B factors to filepath bfactor_comparison_image

# Alternatively, for a more interactive experience Jupyter Notebook can also be used
# See /ANMUtils Setup example for more information on setting up a kernel
# and further usage in Jupyter notebook

# Assuming the B factors match close enough for your system of interest,
# we now export our simulation files
# For all models, this consists of a single function call
#that uses the model itself as its main argument

m.export_to_simulation(our_anm, pdbfile)

# Generates Topology, Configuration, and Parameter File for system
```