Aksimentiev Group
Department of Physics and
Beckman Institute for Advanced Science and Technology
University of Illinois at Urbana-Champaign

Analyzing the Changes in DNA Flexibility Due to Base Modifications Using NAMD

Authors:
Jejoong Yoo
James Wilson
Aleksei Aksimentiev

Contents

1 References

- Modifications of CpG cytosine:
  Thuy Ngo, Jejoong Yoo, Q Dai, Q Zhang, C He, Aleksei Aksimentiev, and Taekjip Ha, Effect of Cytosine Modifications on DNA Flexibility and Nucleosome Mechanical Stability. Nature Communications 7 10813 (2016) DOI:10.1038/ncomms10813

- Modifications of uracil:
  Spencer Carson, James Wilson, Aleksei Aksimentiev, Peter R. Weigele, and Meni Wanunu, Hydroxymethyluracil modifications enhance the flexibility and hydrophilicity of double-stranded DNA. Nucleic Acids Research 44(5) 2085-2092 (2016) DOI:10.1093/nar/gkv1199

- Ion parameters for MD simulation of DNA:
  Yoo, J. & Aksimentiev, A., 2012, Improved Parametrization of Li+, Na+, K+, and Mg2+ Ions for All-Atom Molecular Dynamics Simulations of Nucleic Acid Systems, The journal of physical chemistry letters, 3(1), pp. 45–50. DOI:10.1021/jz201501a

2 Introduction

In this tutorial, we walk through the protocol for all-atom simulations of a double-stranded DNA helix with chemical modifications using the NAMD package. NAMD and VMD programs and minimal experience on those programs are required to follow this tutorial. See the following links for the tutorial of NAMD and VMD programs:

- The NAMD program: http://www.ks.uiuc.edu/Training/Tutorials/namd/namd-tutorial-unix-html/index.html
- The VMD program: http://www.ks.uiuc.edu/Training/Tutorials/vmd/tutorial-html/index.html
- AMBER tools (http://ambermd.org/#AmberTools)
  or 3D DART (http://haddock.science.uu.nl/dna/dna.php)
- x3DNA (http://x3dna.org)

This tutorial is accompanied by prewritten script files for NAMD simulation inputs and analysis: `dnaTutorial.tar.gz`.

$$\text{dnaTutorial (working directory)} \begin{cases} \text{toppar: CHARMM36 force field} \\ \text{MDfiles: Scripts for MD simulations} \\ \text{ANALYSISfiles: Scripts for analysis} \end{cases} \quad (1)$$

3 MD simulations of a double-stranded DNA (dsDNA) helix

Here, we build an MD simulation system with a 30-basepair dsDNA helix in a 1-M KCl solution. We change the sequence, or add chemical modifications to the DNA and perform MD simulations of the system. All script files can be found in MDfiles folder. MDfiles/do.sh includes a shell script that performs the entire procedure in this section.

3.1 Building the structure of a dsDNA helix

Unlike protein simulations for which one usually starts MD simulations using an experimentally determined native structure, the native structure of a given DNA sequence can rarely be found except for several well known DNA sequences. Thus, we build a dsDNA helix with a canonical B-DNA conformation using the Nucleic Acid Builder (NAB) program in AMBER tools. The NAB program requires a simple input file that specifies the sequence of the DNA and the form of the DNA. As an example, we provide the file, MDfiles/ambertools.nab, which can be used to construct a double-stranded DNA and contains the following:

```
// Program 1 - Average B-form DNA duplex
molecule m;

m = bdna( "acagtcatcgcgcgcgcgcgttagctcg" );
putpdb( "nab.pdb", m );
exit( 0 );
```

the bdna command directs the building of a dsDNA helix of the given 30-bp sequence and the putpdb command writes the dsDNA helix into a PDB file. Note that we have a “tacg” motif in the center. Later, we can chemically modify the CpG cytosines or TpA thymines. Once the NAB input file is created, one can execute the program:

```
export NABHOME=/path/to/ambertools/dat/
export AMBERHOME=/path/to/ambertools/
export PATH=$PATH:$AMBERHOME/bin

nab -o nab.out ambertools.nab
./nab.out
```

For convenience, the environment variables (the first three lines) can be added to the user's .bashrc or .profile file.

Alternatively, one can use the 3D-DART web server to build a dsDNA structure: http://haddock.science.uu.nl/dna/dna.php.

3.2 Building the dsDNA molecules for MD simulations

Now, we have the structure of a dsDNA helix in the file nab.pdb. Using this PDB file, we can create the input files for NAMD simulations (PSF and PDB files) using MDfiles/psfgen.tcl.

We must first split the nab.pdb file into nab1.pdb and nab2.pdb, each of which contain one strand (this is due to the requirement of psfgen – the program that prepares the bonding and angles, etc information for simulation – to work on a single chain at a time). We will use the unix grep command for this:

```
/bin/grep ' A ' nab.pdb  > nab1.pdb
/bin/grep ' B ' nab.pdb  > nab2.pdb
```

Then, we create PSF and PDB files (that are consistent with each other) using the MDfiles/psfgen.tcl script. You can execute this script in the shell by typing in the following:

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

To better understand this process of generating a psf file, we describe the process below (skipping some of the housekeeping portions of the script – have a look at psfgen.tcl to see these other pieces). In psfgen.tcl, we load nab1.pdb and nab2.pdb into VMD:

```
for {set i 1} {$i <= 2} {incr i} {
    mol load pdb nab${i}.pdb
```

For each strand i, we build a segment with segid Di:

```
    segment D${i}  {
        first 5TER
        last 3TER
        pdb nab${i}.pdb
    }
```

In this tutorial, we use the CHARMM36 force field. By default in CHARMM36, nucleic acids are in the RNA form. To convert from RNA to DNA, we deoxidize each residue by applying DEO5 or DEOX patch. Note that DEO5 is used for 5-terminal nucleotide, and DEOX for the rest. Additionally, we apply patches for chemical modifications that are defined in the toppar/toppar_all36_na_modifications.str file. For cytosine, we have five patches: methylcytosine (5MC2), hydroxymethylcytosine (5HMC), formylcytosine (5FC2), carboxylcytosine (5CAC), and protonated carboxylcytosine (5NCC). For thymine, we have two patches: hydroxymethyluracil (5HMU) and phosphorylated hydroxymethyluracil (5PMU). Note that 5MC2 is included in the CHARMM36 force field by default, and the others are created using the ParamChem.org website. For example, we can modify CpG cytosine nucleotides (the 9th, 15th and 19th nucleotides of each strand) to methylcytosine by applying the 5MC2 patch.

```
    foreach resid [join [lsort -unique [$sel get resid]]]  {
        if {$resid == 1} {
            patch DEO5 D${i}:$resid
        } else {
            patch DEOX D${i}:$resid
        }

        if {($resid == 9 || $resid == 15 || $resid == 19)} {
            patch 5MC2 D${i}:$resid
        }
    }
```

By creating segments and applying patches, we have completed supplying the information for the *topology* – the atom types, bonds, angles, dihedrals, etc. Finally, we load coordinate information to the segments:

```
    coordpdb nab${i}.pdb D${i}
}
```

At the end of MDfiles/psfgen.tcl, we write PSF and PDB files into a single dna.psf and dna.pdb, respectively:

```
writepsf dna.psf
writepdb dna.pdb
```

3.3 Building the simulation box of an explicit ionic solution

Using MDfiles/solvate.tcl, we add the dsDNA helix in a 1-M KCl solution. The shell command for this is:

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

In MDfiles/solvate.tcl, we execute two commands, solvate and autoionize, which adds a water box and adds ions, respectively. The solvate command

```
solvate dna.psf dna.pdb -minmax {{-25 -25 -65 } { 25  25  65 }} -o dna_W
```

reads dna.psf and dna.pdb files, and puts the dsDNA structure in a water box of 50×50×130 $Å^3$ centered at the origin. If you use a different length of DNA, you can change the box size to make sure that DNA has sufficient padding of water around it. The new solvated system will be stored in PSF and PDB files, which are named dna_W.psf and dna_W.pdb.

Then, the autoionize command

```
autoionize -psf dna_W.psf -pdb dna_W.pdb -sc 1.0  -cation POT -o dna_WI
```

reads dna_W.psf and dna_W.pdb files, and randomly places potassium and chloride ions to achieve a 1 M concentration. The new PSF and PDB files will be written to dna_WI.psf and dna_WI.pdb files.

3.4 Equilibration using molecular dynamics

When the water and ions are placed, they are placed randomly, and there may be high energy clashes that would apply very large forces at the beginning of any subsequent simulations. The DNA double helix is fairly fragile, and it is possible that hydrogen bonds could be broken early on in the simulations because of these large forces that do not correspond to experimental conditions. Therefore, we apply restraints to the DNA to hold everything in place while

the water molecules and ions have a chance to adjust. The following script will write out the initial positions of the DNA atoms, so that we can restrain the DNA to its initial coordinates.

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

Additionaly, an implicit extra bond is added between the central nitrogen-nitrogen atoms of the terminal base-pairs, Figure 1, to prevent the DNA ends from fraying during the simulation. The following of the syntax of the extra bond file provided as MDfiles/extrabonds.txt. Each line contains a single bond with the indices of the two atoms, the bond strength (in kcal/mol/Å^2), and equilibrium distance (in Å). The pairs of indices in extrabonds.txt need to be replaced by actual atom indices obtained from a VMD session.

```
bond 888 916 1 2.87
bond 1786 18 1 2.90
```

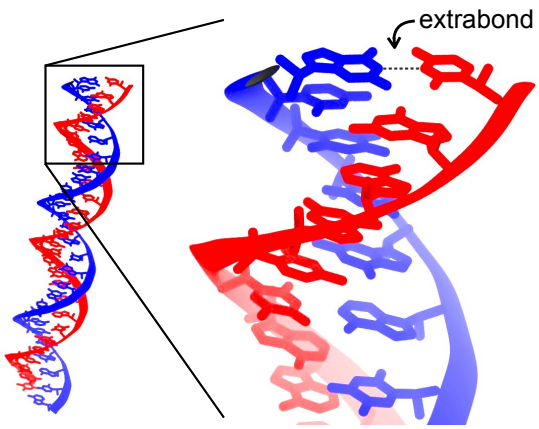

Figure 1: The dashed line indicates the location of an extra bond which prevents the DNA ends from fraying during the production simulation

Next, we will perform an energy minimization process to decrease the effect of atoms being randomly placed too close or too far from each other. This simulation won't take very long (a few minutes) and can be performed on a modest workstation easily.

```
namd2 min.namd > min.log
```

Next, we will perform a short simulation at constant volume and a specified temperature. Sometimes if we immediately start a constant pressure simulation after minimization, the volume will change too rapidly, causing the simulation to crash. This simulation will prevent that from occurring, but will be run at a pressure not consistent with experimental conditions. We will continue to restrain the DNA to its initial coordinates. This will simulate around 1 ns of dynamics and will take some time (about 6 hours on a quad-core workstation without a GPU).

```
namd2 equilNVT.namd > equilNVT.log
```

Next, we will perform a simulation at constant pressure and temperature, while still restraining the DNA to its initial coordinates. This simulation will take approximately five times as long to complete as the previous simulation.

```
namd2 equilNPT.namd > equilNPT.log
```

Finally, we will perform the production simulations in which we remove the restraints from the DNA. The DNA will need to relax for some time after removing the restraints, at least 10 ns. To get meaningful results, this simulation will need to be run for about 40 ns, which will take around 1-2 days using a high-end workstation with a high-end GPU (GTX 1080 or better). Make sure you are using a build of NAMD that can utilize the features of your workstation, i.e. use the multicore or CUDA builds if you have that functionality, and make sure to add the +p, and +devices switches to the command when using more than 1 processor or GPU. The configuration file, product.namd, can be run with the following command for this final phase.

```
namd2 product.namd > product.log
```

4 Analysis of MD trajectories

After the simulations are completed, the flexibility of the DNA double helices containing substitutions and modifications can be compared to unmodified DNA using the x3DNA program. The x3DNA program can be downloaded from x3dna.org (this will require registering on the website). The program does not allow input from DCD files directly, so we must follow several steps to prepare the output of our simulations in a format that x3DNA can use. By way of example, we will be analyzing an MD trajectory of native, unmodified,

DNA that is stored in the archive included in this tutorial in the directory MDfiles/5000. Replace the names of these files with the files for the simulations you wish to analyze.

4.1 Preparing x3DNA input files

First, to make things easier, we will strip water from our DCD files. This allows the dcd to take up less space, and will also facilitate making a movie of the DNA as well. We will also center the DNA in the simulation cell and make sure that the two DNA strands haven't been separated due to simulation cell wrapping issues. To strip water, we will first make a psf and pdb without water. We will then make an index file and feed the dcd and the index file into catdcd, which will produce a new dcd without water that takes up much less space.

The vmd script, strip.tcl, accomplishes these tasks and takes as input a psf file, a dcd file, and an output directory. Enter the following commands into your shell to strip the water from the simulation you have run

Continuation of lines. In the code samples below, a backslash at the end of a line is used to denote a continued line. In most shell environments, a backslash followed by the return key continues a line, allowing the entire line to fit in your screen without wrapping. Depending on your PDF viewer, you should be able to copy and paste the commands in the boxes directly into your command line. However, you can also type them in, either hitting the enter key after the final backslash in a line, or omitting the backslash.

```
vmd -dispdev text -e ANALYSISfiles/strip.tcl \
-args MDfiles/5000/dna_WI.psf  MDfiles/5000/run_WI.dcd native
```

We will be using the output directory "native" to store the analysis. This will create the pdb, psf, and dcd files native/dna_WI.str.psf, native/dna_WI.str.pdb and native/run_WI.str.dcd.

Because x3DNA does not read DCD files, we need to create a pdb file for every frame in the trajectory of the simulation. This is accomplished in vmd by creating a for loop over all frames in the DCD file and performing a writepdb command for each frame.

```
vmd -dispdev text -e ANALYSISfiles/writepdbs.tcl \
-args native/dna_WI.str.psf native/run_WI.str.dcd native/dna.
```

In this example, we chose native/dna. as the prefix for the output pdb files, making native/dna.XXXX.pdb the names of the pdb files, where XXXX is the frame number.

Once the pdb files have been created, they can be combined into a single pdb file by using the unix command *cat* to concatenate all of the files. After they are in a single file, each pdb within the larger file needs to be set off by MODEL and ENDMDL keywords. this can be accomplished by the following awk and sed command:

```
cat native/dna.{16999..19999}.pdb > native/combined.pdb

awk '{if ($1=="CRYST1"){n = n+1;print "MODEL",n} else print}' \
native/combined.pdb | sed -e  's/END/ENDMDL/g' > total.pdb
```

NOTE: While it is possible (and easier) to create a single pdb for the entire simulation, the x3dna_ensemble program is not optimized for very large files and will get VERY slow if we feed it a file containing 10,000 frames. For large simulations with many frames, several pdb chunks should be created to speed up the process. Look in the script, ANALYSISfiles/runx3.sh to see how to accomplish this.

NOTE: For the next two commands to succeed, the x3dna program must be installed; i.e. the X3DNA environment variable must be set and PATH must include the x3dna bin directory.

The x3dna program must know which bases are paired together to complete its analysis. To give this information to the program to inform the analysis, we must create a file that catalogs the basepairs that are in the DNA that we are working with. It is best to create this file from the original canonical DNA that we started our simulation from, as the software can get confused if the bases aren't in correct alignment, which is possible to occur during the simulation. To make this easier, however, we will use the stripped pdb file we created above as the input to create the pair file.

```
find_pair native/dna_WI.str.pdb native/dna.bps
```

4.2 Running x3DNA and parsing the output

Now we will run the x3dna command that will analyze structural parameters for each base. This will take the most time to complete. It is very much recommended to break this portion of the analysis into smaller chunks if analyzing more than 3000 frames.

```
x3dna_ensemble analyze -b native/dna.bps \
-e native/total.pdb -o native/dna.x3
```

This produces a text file "native/dna.x3" in our example, which contains 66 blocks of data. Each block represents one parameter that is measured from our input pdb file, and contains as many lines as frames in our original dcd file. The first column denotes which frame is being measured, and the nth column contains the measurements for the nth base pair or nth step. Even though 66 parameters are measured, only 12 of them are relevant for our analysis of the flexibility of DNA: shear, stretch, stagger, buckle, propeller, opening, shift, slide, rise, roll, twist, and tilt. The first six of these parameters describe the structure of a single base pair, and the last six describe the structure of a step (two adjacent base pairs).

The file structure of the output of the x3dna analyze program is not very compact, and so in an effort to save space, and avoid having to read large text files repeatedly while debugging our analysis, we will repackage the data from the 12 parameters of interest into a binary numpy archive. The included x3dna.py header file contains functions for reading x3dna files, packing and unpacking the data. If you do not have a PYTHONPATH set in your environment, create a directory to hold your python header files, and add the path to your PYTHONPATH environment variable by editing your /.bashrc file to include the following line:

```
export PYTHONPATH=/Path_to_directory_just_created:$PYTHONPATH
```

Then copy x3dna.py from ANALYSISfiles to a directory in PYTHONPATH. NOTE: you must have numpy and matplotlib installed into your python distribution to use the header file. Consult with your system administrator to install the packages, or download anaconda (anaconda.org) and follow the directions for installing python packages. If you have administrator priveleges in Ubuntu linux, you can install these packages by executing the following command:

```
sudo apt-get install python-numpy python-matplotlib
```

Once the header file is installed, we can execute the following python program to process the x3dna analysis files.

```
python ANALYSISfiles/processx3files.py native native/dna.bps \
native/native.npz 0.0024 native/dna.x3
```

Which will create a file native/native.npz that contains all of the information needed to analyze the flexibility of the DNA contained in a compressed numpy file. At this point, the *.pdb and *.x3 files can be safely deleted, as they are $\sim$200X larger than native/native.npz, and the information we need is contained in native/native.npz.

NOTE: This npz file can be read using the function readShortNPZ() in the x3dna.py module that is supplied along with this document. It takes the name of an npz file as an argument, and returns a list of five items:

- **data, keys, bps, name, dt**
- **data** The first item is a numpy array of shape (12,T,D) with all of the time-series data in it, where 12 is the number of structural parameters, T is the number of time steps, and D is the number of DNA bases in the analyzed DNA.
- **keys** The second item in the list is a list of the twelve names of the x3dna parameters we are analyzing. These twelve names correspond to the first dimension of the data array e.g. data[0] will contain the data for the parameter with name keys[0].
- **bps** The variable "bps" contains a list of the names of the basepairs in the DNA, and is length D (defined above).
- **name** is a simple name for the simulation, and is input when processx3files.py is invoked.
- **dt** Finally, dt is the time between frames in the analysis, in units of nanoseconds, and is helpful in constructing time-series graphs.

4.3 Analyzing DNA flexibility

Next, we will plot the DNA structural parameters over time to make sure that the assumptions of flexibility are met in the simulations we performed.

We assume that the value of each structural parameter is distributed about a mean value, and that there is a local harmonic potential, with a minimum that is also centered at that mean value. By measuring the standard deviation of the structural parameter, we can guess the strength of the underlying harmonic potential. Therefore, if the standard deviation of a parameter, say "roll", is larger in DNA fragment A than in DNA fragment B, we can deduce that DNA fragment A is less confined than DNA fragment B, and thus less rigid than DNA fragment B. This deduction is based on the assumption that the DNA is

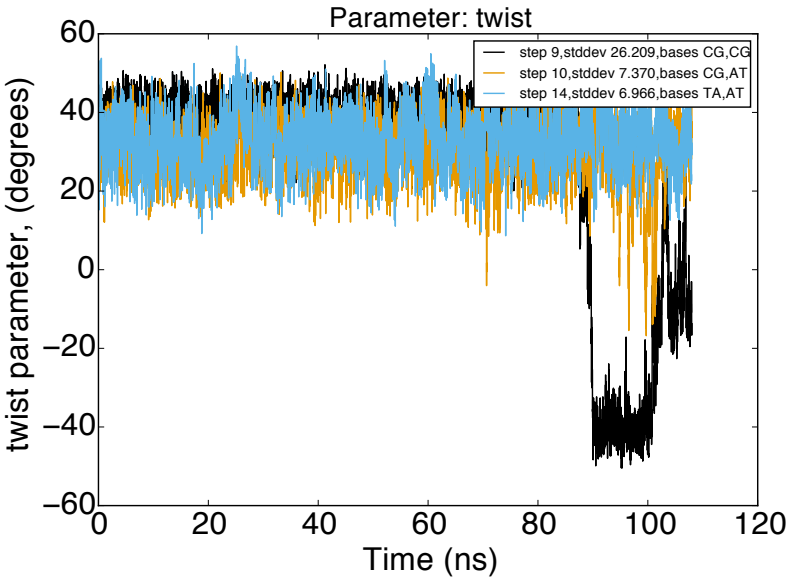

Figure 2: Plot of twist parameter (for the three steps with the largest standard deviation of twist) over the course of a 100 ns simulation. A jump in the twist parameter for step 9 occurs at $\sim$85 ns, settling on a new equilibrium configuration for around 10 ns.

near equilibrium around a mean value. However, if a base pair or step transfers from one local minimum to another during a simulation, the standard deviation would underestimate the strength of the potential well. For each of the structural parameters, we will look at the base pairs or steps with the largest standard deviations and see if the parameter is mostly clustered around a single equilibrium value, or if it makes large jumps into multiple local equilibria.

For example, Figure 2 shows an example of a single step moving into a different equilibrium configuration. For step 9, the twist parameter changes from an average value of around 40 degrees to -40 degrees at around 85 ns. When these parameters change during the simulation to be centered about new mean values, the standard deviation of the parameter no longer reports accurately on the near-equilibrium flexibility of the DNA. Therefore, if planning on measuring the flexibility of the DNA near step 9, the data after 85 ns would need to be discarded.

To run this analysis and check the stability of the DNA you simulated earlier, run the following command:

```
python ANALYSISfiles/examinex3.py native/exam native/native.npz
```

This command will create twelve pdf files with names such as native/exam.twist.pdf, and native/exam.roll.pdf. Open these files in a pdf reader to make sure that there are no large jumps in any of the parameters during the simulation.

Next, we will take a look at how the flexibility of the DNA changes along its length. DNA usually exhibits increased flexibility near the ends of a DNA fragment, and, depending on the contained bases and neighbors, a base pair or step will feature different flexibility. Run the following command to create figures of the flexibility of DNA as a function of basepair (or step) number.

```
python ANALYSISfiles/flex3.py native/flex native/native.npz
```

Again, this command creates twelve pdf files with names starting with native/flex. Open these files to see the average standard deviation of each base in the native sequence. Many of these graphs of the standard deviation of structural parameters show a periodic variation, with relatively stiff base pairs (or steps) next to relatively flexible base pairs (or steps). The error bars on these graphs are calculated by breaking the time series of the structural parameter data into 30 bins, calculating the standard deviation in each of these 30 bins, and then calculating the standard error of these 30 standard deviation estimates.

At this point, we have created estimates of the fluctuation of structural parameters of DNA molecules, however, these estimates are generally useless

unless compared to a reference flexibility. We want to know if a particular modification or change in sequence can affect the flexibility of a DNA molecule. To do this, complete the above steps for another DNA variant, and when that is done, we will make a comparison between the two in the next section.

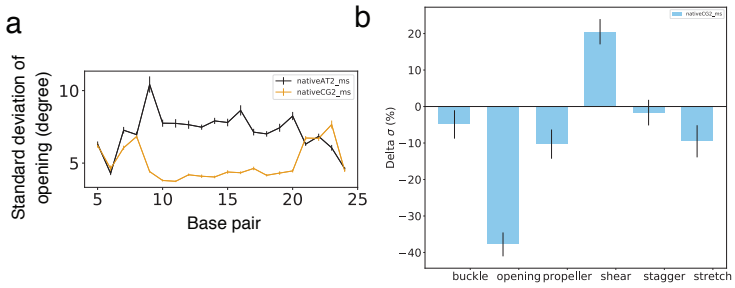

Figure 3: a. The standard deviation of the opening parameter for each base on the DNA strand. b. Comparison between the standard deviation of several parameters between AT rich and CG rich strands.

4.4 Comparing flexibility of DNA variants

First, to get a general picture of how a modification or change in sequence affects stability, we will compare an A-tract of DNA with heterogeneous sequence DNA. Once both simulations and the above analysis steps are complete, input the following command.

```
python ANALYSISfiles/comparex3.py AAAA/comp \
native/native.npz AAAA/AAAA.npz
```

This will perform the analysis we performed with flex3.py in the previous section for both the native DNA and the DNA with a sequence of AAAA embedded in it, and plot the flexibility of both DNA on the same graph, Figure 3a. The remaining bases in both DNA are identical, so you should notice the standard deviation of the basepairs and steps should be close for all bases far from where the AAAA sequence is in the A-tract. Also, you should see that the characteristic alternating stiff-flexible pattern, which is present for many parameters in the native sequence, has been suppressed in the AAAA sequence, replaced with a fairly constant low-flexibility pattern.

While the above comparison is a good check, the neighboring sequences can add variability to the modified region causing some bases to be more flexible, and others to be less flexible. To get the overall effect of the modifications, you can calculate the percent change in flexibility and average over all modified

bases, Figure 3b. To calculate the percent change in flexibility, you can use the following equation.

$$\Delta\sigma = \frac{\sigma_r - \sigma_m}{\sigma_r} \cdot 100 \tag{2}$$

Where $\Delta\sigma$ is the percent change in flexibility for a given parameter, and σ_r, and σ_m are the standard deviation of the reference and modified DNA for that parameter respectively. We then take the mean over all affected basepairs and steps. To calculate the uncertainty for $\Delta\sigma$, you can propagate the uncertainty using the usual methods (for example wikipedia has an article on propagation of uncertainty). We have included a python program to calculate and plot $\Delta\sigma$ along with the error bars. To use this program, input the following command:

```
python ANALYSISfiles/comparePercent.py AAAA/percent \
native/native.npz AAAA/AAAA.npz
```

The program will create two files: AAAA/percent.intra.pdf, and AAAA/percent.inter.pdf. The first file will have bar graphs of the percent change in flexibility for the intra-DNA structural parameters, and the second will have the same for inter-DNA structural parameters. If the majority of values are negative, it shows a decrease in flexibility, and if the majority of values are positive, it shows an increase in flexibility.

5 Troubleshooting

5.1 When equilibration simulations crash

At the beginning, the system can be unstable. You need to check total energy. If total energy looks fine and your system is still unstable, you might try:

- Decrease timestep
- Turn off pressure coupling; do NVT.

Somehow both genbox (gromacs) and solvate (VMD) put significantly too few water molecules in a box, resulting in bubbles in the box after equilibration in NVT.

Solutions:

1. Resolvation after equilibration.
2. NPT with DNA fixed.

When a DNA origami object is large, the number of inserted water molecules can be underestimated, causing a bubble after a short equilibration. Bubbles of moderate size can be removed by simply equilibrating under NPT ensemble. Sometimes large bubbles can abrupt change in the box size, resulting in instable MD simulations. In this case, increasing langevinPistonPeriod and langevinPistonDecay by a factor of 10 can be helpful. By changing these options, the box size is less responsive to the internal pressure, making the MD simulation more stable. After the box size becomes stabilized, those options can be reverted to the default values.