Proceedings of the 2023 Winter Simulation Conference
C. G. Corlu, S. R. Hunter, H. Lam, B. S. Onggo, J. Shortle, and B. Biller, eds.

COARSE-GRAINED SIMULATIONS OF DNA AND RNA SYSTEMS WITH OXDNA AND OXRNA MODELS: TUTORIAL

Matthew L. Sample

Fulton School for Engineering
of Matter, Transport, and Energy
Arizona State University
Tempe, AZ 85281, USA

Michael Matthies
Petr Šulc

School of Molecular Sciences
Arizona State University
Tempe, AZ 85281, USA

ABSTRACT

We present a tutorial on setting-up the oxDNA coarse-grained model for simulations of DNA and RNA nanotechnology. The model is a popular tool used both by theorists and experimentalists to simulate nucleic acid systems both in biology and nanotechnology settings. The tutorial is aimed at new users asking "Where should I start if I want to use oxDNA". We assume no prior background in using the model. This tutorial shows basic examples that can get a novice user started with the model, and points the prospective user towards additional reading and online resources depending on which aspect of the model they are interested in pursuing.

1 INTRODUCTION

The fields of DNA and RNA nanotechnology use designed nucleic acid strands to self-assemble nanoscale structures and devices. In the past forty years since the ground-breaking work by Ned Seeman that has established the field (Seeman 1982), structures of increasing complexity and sizes have been experimentally realized. The promising applications of the field include diagnostics, therapeutics, molecular computation and templating for nanoscale assembly (Seeman and Sleiman 2017; Pinheiro et al. 2011). One of the most commonly used constructs in the DNA nanotechnology field is DNA origami (Rothemund 2006). DNA origami consists of a long (7000 bases) single-stranded DNA scaffold strand, and shorter (order of tens of nucleotides) staple strands that connect different domains together. Computer simulations can provide important information into the function of these devices, but also presents multiple challenges. The substantial system sizes (encompassing up to tens of thousands of nucleotides) as well as the occurrence of rare events that may be crucial to the function of the device (e.g. breaking and creation of base pairs) present significant modeling challenges. Hence, coarse-grained models, where each particle represents multiple atoms (Sengar et al. 2021; Maffeo and Aksimentiev 2020), are often employed as opposed to models with atomistic-resolution.

Among the leading coarse-grained models for DNA and RNA nanotechnology are oxDNA and oxRNA (Ouldridge et al. 2011; Šulc et al. 2012; Šulc et al. 2014; Snodin et al. 2015), which have been parametrized to reproduce basic structural, thermodynamic and mechanical properties of DNA and RNA. Where available, the models have been shown to be in good agreement with available experimental data. They have been utilized for nanostructure design and verification, as well as for studies of active DNA devices such as walkers, as well as for biophysical properties of DNA and RNA (Doye et al. 2013; Sengar et al. 2021). In this tutorial, we describe the basics of the models, and offer a walk-through for setting up a simulation. It is meant for users who are new to nucleic acid nanotechnology modeling and are wondering

what are the good resources to get started. For a more exhaustive understanding of the oxDNA modeling tools ecosystem, we direct readers to additional references.

2 THE OXDNA MODEL AND WHEN TO USE IT

OxDNA/oxRNA are coarse-grained models are highly coarse-grained, with 1 rigid body (bead) representing the entire nucleotide (see Fig. 1). The beads have multiple interaction sites, and the interaction between the sites are empirically parameterized to reproduce DNA and RNA hairpin and duplex melting as given by the SantaLucia and Turner nearest-neighbor models (SantaLucia Jr 1998; Mathews et al. 1999). The models also capture the transition from single strands to duplex or stem formation, and represent the mechanical properties of highly flexible single-strands, while duplex regions behave as extensible worm-like chains, in good agreement with known mechanical properties of DNA and RNA (Ouldridge et al. 2011; Šulc et al. 2014; Snodin et al. 2015). The schematic illustration of the oxDNA model and its interactions is shown in Fig. 1. More recently, we also introduced the ANM-oxDNA model, which represents proteins alongside the oxDNA or oxRNA model. (Bohlin et al. 2022; Procyk et al. 2021).

While detailed description of the model's parametrization and their properties of provided elsewhere (Ouldridge et al. 2011; Snodin et al. 2015; Šulc et al. 2014; Šulc et al. 2012), the novice user should be aware of the following: 1) The oxDNA model comes in two versions, oxDNA1 and oxDNA2. The latter, oxDNA2, is the recommended choice as it includes salt effects using the Debye-Huckel potential and accounts for major and minor grooving, which oxDNA1 did not capture. Similarly, the oxRNA2 also includes the Debye-Huckel potential as opposed to oxRNA1, which did not support it, and hence is also the recommended interaction to use. 2) Both models can be used with two options: sequence-independent and sequence-dependent. The sequence independent version is the default version. While only A-T (A-U) and C-G base pairs can form in the sequence-independent model, their strength is the same: The model uses averaged value for all of them. This model is suitable for studies where one does not want the results to be occluded due to a particular choice of interaction strength, or where the sequence effects average out due to similar frequency of use of AT and GC base pairs. The sequence dependent model uses stacking and base pairing interaction strengths parameterized to the melting temperature predictions of duplexes. Additionally, in the oxRNA model, it also allows formation of wobble (G-U) base pairs. 3) OxDNA is the name of the coarse-grained DNA model, but is also used interchangeably to refer to the oxDNA simulation package (Poppleton et al. 2023), which is a molecular simulation software package. Additionally, the oxDNA and oxRNA models have also been implemented in the general-purpose LAMMPS simulation package (Henrich et al. 2018).

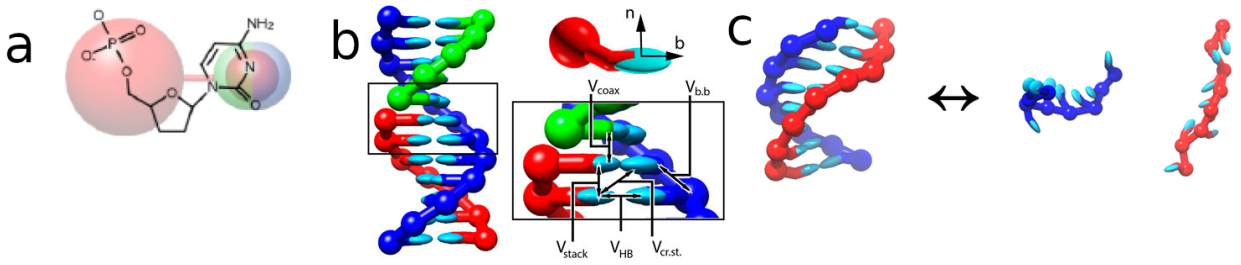

Figure 1: A schematic overview of oxDNA and oxRNA models. a) A superposition of coarse-grained and atomistic nucleotide representations. Even though for visualization purposes we draw the hydrogen bonding and stacking base site as an ellipsoid, the model represents them as spherical excluded volume sites. b) A schematic of a DNA duplex, along with a representation of a single base with position vectors **b** and **n**, as well as a schematic illustration of the interactions in the model (Debye-Huckel repulsion between backbone sites is not shown). c) A schematic of a melting transition in the oxRNA model.

Prior to starting any simulations,it is important to consider if the oxDNA tool can provide valuable insight into the problem you are studying. Namely, one should ask if the same research problem can be answered with a simpler and faster method, and if the process one is interested in is accurately captured by the model, or would answering it require a more detailed / accurate description of the system? Often, one can use secondary structure prediction tools such as ViennaRNA or NUPACK (Zadeh et al. 2011; Lorenz et al. 2011) for fast enumeration of possible interaction states. As opposed to these tools, oxDNA provides a three-dimensional representation, can be used to study kinetics, and simulation of complex 3D structures including pseudoknots. On the other hand, the model lacks atomistic-level resolution and cannot represent interactions with other biomolecules, which limits its applicability to studies of complex biological environments.

Currently, OxDNA and oxRNA are the only models for DNA and RNA nanotechnology that can correctly and efficiently capture single-stranded to double-stranded transitions. This makes them especially useful to study systems where base-pair creation or breaking can occur, as well as for structures which consist of both single-stranded and double-stranded regions. The model has been shown to be able to study systems consisting of up to 1 million nucleotides (Rovigatti et al. 2022). However, simulations of diffusion-limited processes that rely on the diffusion of single-stranded DNA or RNA, can be quite time-consuming. Currently, the model is not fast enough to e.g. simulate assembly of DNA origami structure from individual strands, and advanced biasing methods and significant computational resources need to be applied to study systems of such sizes (Snodin et al. 2016).

The simulation package implements both CPU and GPU molecular dynamics engines, and CPU implementations of the Monte Carlo and cluster Virtual Move Monte Carlo (VMMC) algorithms. The latest version of the code is freely available at https://github.com/lorenzo-rovigatti/oxDNA. The online repository features code documentation, which includes installation instructions and also provides a more comprehensive explanation of the model and its parameters. In the following, we walk through the steps of setting up and evaluating an example simulation. It is beyond the scope of this tutorial to cover all aspects of molecular simulations, and for the general techniques and explanation of theory behind molecular simulation, we recommend popular textbooks such as (Frenkel and Smit 2001; Tuckerman 2010). A more condensed introduction to molecular simulation algorithms is also provided in (Munsky et al. 2018).

3 SETTING-UP AND RUNNING SIMULATIONS

3.1 Creating the Starting Configuration

The first step in setting-up a simulation is the preparation of a system in the oxDNA model file format. The oxDNA (or oxRNA) models requires two files: the topology file (typically ending with .top suffix) and configuration file (typically saved as a .dat file). Below, we list an example of the topology file for a system consisting of two DNA strands, each of length 3 bases:

```
6 2
1 T -1 1
1 A 0 2
1 C 1 -1
2 G -1 4
2 T 3 5
2 A 4 -1
```

where the first line indicates there are 6 nucleotides in two different strands. Then, the sequence is listed in the $3'$ to $5'$ order, with one nucleotide per line. The first column is the id of the strand the nucleotide belongs to, then the base type, and the id of the $3'$ and $5'$ neighbor. Alternatively, it is possible to replace the base type (A,T,C,G) by a number with absolute value greater than 10, in which case it can only bind to a base with id such that their sum is 3. For example, in the following topology the first nucleotide can be

only able to bind to the last one, and no other base pairs are possible for it, thus avoiding some possible undesired mismatches that you want to prevent from forming:

```
6 2
1 13 -1 1
1 A 0 2
1 C 1 -1
2 G -1 4
2 T 3 5
2 -10 4 -1
```

Future stable release of the oxDNA tool will also support a new topology format where the sequence is simply listed as one strand per line, but the above format will remain supported for backward compatibility. An example of the first three lines of the configuration file are

```
t = 0
b = 20.0 20.0 20.0
E = -1.27  -1.42 0.14
```

where the first line is the timestep of the simulation (starting at 0), the second line is the dimensions of the simulation box edges, and the third line is the total, potential, and kinetic energy per particle respectively. It is advised that the box size is large enough such that the simulated structure cannot interact with itself (e.g. has to be larger than the largest distance between any pairs of nucleotides in the system). This can be easily verified by visualizing the structure in oxView tool (Bohlin et al. 2022) and switching on the Box option in View tab. Then, after the top three header lines, the configuration file has one line per particle (not shown above). Each line lists, for each nucleotide in separate columns, x, y and z position, the three components of the particle vectors **b** (pointing from the center of mass of nucleotide to base site) and **n** (perpendicular to **b** and will point towards 5′ neighbor in a formed duplex, see Fig. 1b). Then, the three components of the velocity and angular velocity vector are listed as well. Once the simulation is run, the simulation code will also produce a trajectory file, which is a series of configuration files (in the same format as described above) concatenated one after another.

It would be too tedious to manipulate and create the topology and configuration files directly, so a set of tools are provided to convert from common design tools such as CADNano, Tiamat or PDB format (Douglas et al. 2009; Williams et al. 2009). Some other existing tools, such as MagicDNA or sCADNAno (Huang et al. 2021; Doty et al. 2020), also allow directly saving designed system in oxDNA format.

The standard tool for oxDNA visualization, oxView, is available as a browser-based application at https://oxView.org. It can import from common DNA nanotechnology formats directly into oxDNA, and then save the system as .top and .dat files in the format above. Importing structures from other design tools is advised for origami sized systems, while smaller systems (like tiles) can be created directly using the oxView editing interface, which allows creation and deletions of single-stranded and double-stranded sections and connecting them together. The oxView tool along with its use for structure import and generation is described in detail in (Bohlin et al. 2022). The conversion tool between different formats is also provided as a standalone website http://tacoxdna.sissa.it/ (Suma et al. 2019).

3.2 Installation of the OxDNA Tool

The software tool is intended to be run in a high performance computing (HPC) environment, such as a dedicated workstation or computer cluster. Such systems typically run using Linux systems, and as such, our tool is currently intended to be run in a Linux environment. It can be also run in a Linux virtual system on Windows. To compile the executable (binary) of the program, go to the oxDNA directory after downloading the tool and type

```
mkdir build
cd build
cmake ../ -DCUDA=1
```

```
make
```

If successful, the above command will lead to a binary file called `oxDNA` in the build directory. The above commands also assumes you have the CUDA toolkit library installed on your system that supports GPU computing. If you leave out the `-DCUDA=1` parameter, the code will only be compiled with support for a single CPU simulations. In most HPC settings, one needs to import a NVIDIA CUDA compiler and a compatible C++ compiler (we recommend GNU C Compiler) using the `module add` command. Please refer to your computer cluster documentation for details how to import the right libraries.

Optionally, you can install Python binding libraries as well, which gives you access to the oxpy modules for preparing and running the simulations from a Python interface, as well as providing access to the oxDNA Analysis Tools modules for analysis of simulations in Python. In that case, the installation process is

```
mkdir build
cd build
cmake ../ -DCUDA=1 -DPython=1 -DOxpySystemInstall=1
make
make install
```

The above Python binding will only work with Python version 3.9 or higher. For future reference of troubleshooting of known installation issues, please refer to the online documentation of the tool. For additional information and troubleshooting, follow the instructions on the github repository documentation page https://lorenzo-rovigatti.github.io/oxDNA/.

3.3 Use of Online Resources to Run OxDNA

As an alternative to installing and compiling your own oxDNA code locally, online tools are available to perform simulations for you. To simplify the setup and use of the model, we maintain a free online webserver https://oxDNA.org, where the topology and configuration files can be directly uploaded. The server is equipped with 8 GPU cards and supports oxDNA, oxRNA and ANM-oxDNA simulations. It is meant for users that do not have access to their own computer clusters and are primarily interested in running unbiased simulations of DNA or RNA systems. The details of how to use this service is provided in (Poppleton et al. 2021). For certain applications this might be a better option, as the server interface is optimized for typical use cases of the oxDNA and oxRNA models. It provides basic online analysis tools that can produce mean structures, preform flexibility analysis, create videos of simulated trajectories, show bond occupancy, and calculate distributions of angles between different helices and distances between groups of nucleotides. In the rest of the tutorial, we assume that you are running all simulations on a Linux cluster, and hence need to setup all simulations by yourself.

3.4 Setting up Simulation Parameters

3.4.1 Molecular Dynamics on CPU

Once you have prepared the starting configuration and topology, we need to setup a text file with instructions to run the simulation. This file is typically named `input`, and contains parameters of the simulations. An example of such a file is provided below:

```
####  PROGRAM PARAMETERS  ####
backend = CPU
seed = 4982

####    MD PARAMETERS    ####
sim_type = MD
newtonian_steps = 103
diff_coeff = 2.50
thermostat = john
```

```
dt = 0.003
verlet_skin = 0.05

##### MODEL PARAMETERS #####
interaction_type = DNA2
use_average_seq = no
seq_dep_file = oxDNA2_sequence_dependent_parameters.txt
salt_concentration = 1.0
T = 25C

####    INPUT / OUTPUT    ####
steps = 1e8
topology = 75deg.top
conf_file = 75deg.dat
trajectory_file = trajectory.dat
refresh_vel = 1
time_scale = linear
restart_step_counter = 1
energy_file = energy.dat
print_conf_interval = 1e5
print_energy_every = 1e4
```

Any line that starts with # are treated as a comment and are ignored by the program. The simulation parameters are grouped by their role. First, `backend` specifies we will run the code on CPU, and `seed` initializes the random number generator. Omitting the `seed` parameter prompts the program to default to a randomly generated number when launched. If you run one structure in two different simulations with the same seed, they will produce identical trajectories, so make sure to launch simulations with unique seeds. Next, we set the parameters of the molecular dynamics (MD) algorithm. While detailed description of molecular dynamics is available elsewhere (Frenkel and Smit 2001), it can be approximately understood as numerical integration of Newton's equations of motion. This involves parameters like timestep `dt` and verlet list, which is important for neighbor list calculation, as elaborated in (Frenkel and Smit 2001)). Each nucleotides movement in the simulation is dictated by forces equal to the derivatives of the oxDNA model potential. Additionally, it is coupled with a thermostat, which ensures that the states sampled by the system correspond to the specified temperature. In the example above, we use an Andersen-like thermostat (specified by parameter `john` in reference to the article which introduced it (Russo et al. 2009)). This thermostat periodically resamples random velocities and angular momenta of nucleotides so that they correspond to the Boltzmann distribution at the specified temperature. The parameters of the thermostat are `diff_coeff` which sets the diffusion constant of a single nucleotide, and `newtonian_steps` which is related to the frequency with which the velocities of nucleotides are refreshed. It is advised that these parameters are kept to their values specified above. The `steps` parameters specifies how long the simulation should run (i.e. how many steps it will perform). This number is system-dependent, and it should be as large as necessary to sample all relevant states of the simulated system.

The model parameters section specifies that we use the oxDNA2 model version (`interaction_type = DNA2`), with its sequence-dependent variant. The file `oxDNA2_sequence_dependent_parameters.txt` must either be copied from the main oxDNA directory to the directory where the simulation is launched or the full path to the file location can be provided. The above configuration sets the sodium concentration to 1M using the Debye-Hückel potential, and the temperature to 25 Celsius degrees. An identical set of parameters would also apply to the oxRNA model, however the interaction is called RNA2. Additionally, there is an optional (but highly recommended) parameter `mismatch_repulsion = 1`, which introduces

a weak repulsion between mismatched base pairs to correct for the high stability of mismtached base pairs in the oxRNA model (Šulc et al. 2014).

Finally, the input/output section specifies the name of the topology and starting configuration files (`75deg.top` and `75deg.dat` in this example). The generated trajectory will be saved to `trajectory.dat` (specified to be saved every 10^5 simulation steps). The energy per particle as a function of time will be saved to a file named `energy.dat` every 10^4 steps, using linear time scale. Setting `restart_step_counter` to 1 means that every time the simulation is stated, it deletes the `energy.dat` and `trajectory.dat` files and then restarts the simulation from time 0. If the parameter is set to 0, the simulation will instead continue appending to existing trajectory.dat and energy.dat files, resuming from the time specified in the configuration file. The simulation saves the last configuration when it ends (or the job is cancelled or killed by the server) to a file named `last_conf.dat`. If you use the option `restart_step_counter = 0`, it is vital to ensure you are restarting the simulation from the last simulation state by setting `conf_file = last_conf.dat`.

To run the simulation, it is advised to create a separate directory which contains the topology, configuration and input files. The simulation can then be launched by

```
$OXDNAPATH/build/bin/oxDNA input
```

where $OXDNAPATH is set to the oxDNA installation directory. The program will then run and produce the output files in the same directory. To ensure you simulation is running as intended, it is advised to observe the values of energy printed on the screen (ideally they should be about -1.4 per nucleotide for a system consisting of mostly base-paired nucleotides), and to use oxView to visualize the produced trajectory.

3.4.2 Molecular Dynamics on GPU

Single CPU simulations are usually too slow for most systems of interest. If the size of the studied system exceeds about 300 nucleotides, the GPU simulation can provide up to two orders of magnitude speed-up. To use a GPU-version of oxDNA, the first two sections of the input file have to be changed to

```
####  PROGRAM PARAMETERS  ####
backend = CUDA
backend_precision = mixed
use_edge = 1
edge_n_forces = 1
####    MD PARAMETERS    ####
sim_type = MD
CUDA_list = verlet
verlet_skin = 0.2
max_density_multiplier = 10
CUDA_sort_every = 0
newtonian_steps = 103
diff_coeff = 2.50
thermostat = john
dt = 0.003
```

The detailed explanation and fine-tuning of the parameters above are given in the oxDNA code documentation. For a majority of large systems (e.g. DNA origami), these listed values should provide optimal speed on the latest GPU cards. The remaining sections setting-up the model and input and output files are the same as they were for the CPU implementation.

3.4.3 Monte Carlo Simulation on CPU

The simulation code also supports both the Metropolis Monte Carlo Algorithm and Virtual Move Monte Carlo (VMMC) algorithm for the oxDNA and oxRNA models. The Monte Carlo algorithm relies on proposing a trial move, either with a single particle or a group of particles, and subsequently accepting or rejecting it (Frenkel and Smit 2001). The Virtual Move Monte Carlo algorithm (Whitelam and Geissler 2007) is particularly efficient at sampling the states of smaller systems (ideally less than 100 nucleotides), which it achieves faster than molecular dynamics. The input file for the MC algorithm (which replaces the MD PARAMETERS section of the CPU version of the code) is:

```
####    MC PARAMETERS    ####
sim_type = MC
ensemble = NVT
delta_translation = 0.10
delta_rotation = 0.25
list_type = cells
```

The parameters `delta_translation` and `delta_rotation` set the mean size of proposed translation or rotation moves, and should ideally be chosen so that about 30% of suggested moves are accepted. Large scale moves will be rejected too often, while short moves will be accepted, but overall it will take too many proposed moves before the system significantly changes. Provided `list_type = cells` is one of the possible means (along with verlet lists) that algorithm can efficiently keep track of its neighbors. For the VMMC simulation, the recommended input parameters are

```
####    VMMC PARAMETERS    ####
sim_type = VMMC
ensemble = NVT
delta_translation = 0.10
delta_rotation = 0.25
verlet_skin = 1.0
small_system = 1
maxclust = 30
```

In the above, there are two optional parameters: `maxclust` and `small_system`, which in certain scenarios (typically for system sizes below 50 nucleotides) can increase the speed of the algorithm. When the simulation tool stops, it prints out benchmarking information (such as CPU time per one simulation step) which can be used to fine-tune parameters for maximum efficacy.

3.5 Relaxation of a Structure

When using designs exported directly from design tools, it is often necessary to perform a "relaxation" simulation prior to the production run (the simulation intended to collect data about the system). This relaxation phase is crucial because design tools typically place nucleotides into positions that are not physically possible. For example, adjacent nucleotides on the same strand might be placed too far from each other, exceeding the permissible distance that the covalent bond between nucleotide backbones would allow. The oxDNA model force-field would in such a case result in a nearly infinite force trying to pull these "stretched" nucleotides together, causing numerical instability and program error. Another common issue with imported designs is that sometimes nucleotides are placed too close to each other, resulting in them overlapping. Since the repulsion forces in the oxDNA model will be enormous if two nucleotides are too close to each other, this can result in very high forces and subsequent numerical instability and program crash. It is hence recommended to perform relaxation simulations prior to production runs. We will list the parameters needed to run a relaxation simulation below (they need to be added to either CUDA or CPU simulation input file):

```
####  RELAX PARAMETERS  ####
```

```
dt = 0.001
max_backbone_force = 10
```

Where `dt` is set to a smaller value as compared to the production run to increase numerical stability and `max_backbone_force` is the parameter permitting overstretched backbone bonds. The relaxation steps can be run for a certain number of steps (typically 10^5 to 10^7, but can be longer for larger systems with many stretched regions). In some cases (which feature lots of overlapping nucleotides), it might also be necessary to first run Monte Carlo simulation before proceeding to molecular dynamics relaxation. The automated relaxation is also implemented in the oxDNA.org webserver.

3.6 Running Simulations Using the Jupyter Notebook Interface

Beyond the command line interface, oxDNA also supports python bindings. This capability facilitates both oxDNA simulation and analysis directly from a Jupyter notebook. The setup is described in more detail in the online documentation at https://lorenzo-rovigatti.github.io/oxDNA/oxpy/index.html#an-example-of-a-simple-simulation. To streamline the setup of routine simulations, we provide basic boiler plate code. The Jupyter notebook code provided below preforms the setup of a default production run simulation and provides a convenient way to view and analyse the trajectory.

```
from oxDNA_analysis_tools.UTILS.oxview import from_path
from oxDNA_analysis_tools.UTILS.boilerplate import setup_simulation,\
                                get_default_input,\
                                Simulation
# get default settings for a CUDA simulation
parameters = get_default_input()
# setup sequence dependence
parameters["use_average_seq"] = "no"
parameters["seq_dep_file"] = "../../oxDNA2_sequence_dependent_parameters.txt"
# setup simulation
input_file = setup_simulation("./75deg.top", # topology
                  "./75deg.dat", # starting configuraiton
                  "./simulation",      # output path
                  parameters,  # simulation parameters
                  kill_out_dir = True)

s = Simulation(input_file)
# view initial configuration and run simulation
s.view_init()
s.run()
```

This code will setup and run a production run of the 75 degree layered crossover tile. To monitor the progress of the simulation you can utilize the following code:

```
s.plot_energy()
```

This will produce a plot of the total energy per simulation step. The red line indicates the completion of the simulation. This plotting function can be rerun as many times as needed to follow the simulation progress. To visualize progress one can additionally utilize following code, visualizing the last configuration:

```
s.view_last()
```

Finally after the simulation finishes one can analyze the resulting trajectory using the following jupyter code, calling the oxDNA analysis tools mean script.

```
oat mean ./simulation/trajectory.dat -o ./simulation/mean.dat -d\
                        ./simulation/devs.json
```

The resulting files can be visualized in the Jupyter notebook using:

```
from_path("./simulation/75deg.top","./simulation/mean.dat",
"./simulation/devs.json")
```

We provide this code and the configuration files as a Jupyter notebook in the example folder "BOILERPLATE_Jupyter", shipped with the oxDNA code.

4 ADVANCED SIMULATIONS

The examples above covered the basic setup for unbiased simulations on GPU and CPU. For many systems, this might not be enough: If, even after few days of simulations, you have still not seen the simulated system sample the relevant states often enough, this suggests that the unbiased simulation is not providing sufficient data to be representative of the full behavior of the structure. For example, from experimental FRET measurements, you know that the arms of a DNA nanostructure sample certain ranges of possible distances. However, if the oxDNA simulations do not show the entire range of the motion, it means that it has not been run long enough, or that more advanced methods to enhance the sampling need to be employed. The oxDNA simulation implements discrete umbrella sampling in combination with the VMMC algorithm, which is an ideal tool to sample the entire free-energy landscape as a function of numbers of base pairs in the system. Examples of use include simulations of melting of duplexes, hairpins and pseudoknots and strand displacement system (Srinivas et al. 2013; Hong et al. 2020). This method is documented, along with provided example codes, on the github repository of the oxDNA code. We also support umbrella sampling as a function of continuous parameter (typically distance between groups of nucleotides) that can be used to enhance the sampling of large scale conformational change of different structures. The example code to setup such simulations is available as part of (Sample, Matthew and Matthies, Michael and Šulc, Petr 2023) and is also discussed briefly below. Finally, the model also support enhanced kinetics sampling to extract e.g. rates of a particular reaction such as strand displacement. The forward flux sampling method is supported both on CPU and GPU in the oxDNA code, with examples provided in the the oxDNA documentation on github.

4.1 Continuous Parameter Umbrella Sampling

An open-source Python wrapper of the oxDNA Python bindings, available at https://github.com/mlsample/ipy_oxDNA, provides a semi-automated Jupyter notebook interface to run continuous parameter umbrella sampling. This runs on the GPU implementation of the oxDNA molecular dynamics engine, and as such allows for enhanced sampling of origami-sized systems or larger. The Jupyter notebook interface enables a greater ease of use and supports reproducible and open source simulation notebooks. Furthermore, NVIDIA's multiprocessing service can be harnessed to increase simulation throughput over 2.5x by allowing over 40 origami sized simulations to be run on a single GPU.

Umbrella Sampling works by adding an external biasing potential to the system to enforce sampling all values of the chosen order parameter. An order parameter is any measurable value that is of interest in a simulation. An example of a chosen order parameter can be the distance between two groups of nucleotides on the ends of a DNA origami that we expect to undergo large scale conformational change over a range of distances in the simulation. The biasing potential effectively 'flattens' the energy landscape, making it easier for the system to overcome energy barriers and explore a wider range of states. To ensure we span

the entire range of your order parameter of interest, we run multiple simulations (windows), where each window samples only a specific subset of the order parameter.

The set of biased simulations obtained by running different windows are then combined using the Weighted Histogram Analysis Method (WHAM) (Grossfield, Alan). WHAM effectively 'removes' the bias imposed by the external biasing potential applied during the simulations, and then stitches the distributions of each window together to give a single unbiased estimate of the free energy profile for your order parameter. Careful selection of the order parameter is crucial, as it should be able to effectively distinguish between the different states or conformations of the system. Following the import of the necessary python modules:

```
from umbrella_sampling import ComUmbrellaSampling
from oxdna_simulation import SimulationManager
```

the first step to use the python wrapper is to specify the directory that contains the oxDNA conformation and topology files (file_dir), which is ideally prepared in the command line prior to running the simulation. Next, you need to specify a unique name for the directory which will contain our umbrella sampling system (system). We can then define our order parameter, where in this example we use the distance between the center-of-mass between nucleotides in group one (com) and group two (ref), defined by their nucleotide index.

```
# Define the directory where your files are stored
file_dir = 'abs/path/to/files'

# Define the name for your system
system  = 'user_defined_system_name'

# Define the particles for center-of-mass and reference
com_list = '1,2,3' # This is a comma-separated list of particle indices
ref_list = '4,5,6' # These indices can be found using oxView
```

We next set the umbrella simulation parameters which are determined *a priori* or refined over umbrella simulation trials. These include the order parameter sampling range (xmin, xmax), the number of simulation windows (n_windows), and the stiffness of the external biasing potential (stiff).

```
xmin = 0      #Lowest com distance attemped sampling
xmax = 10     #Highest com distance attemped sampling
n_windows = 20 #Parallelizable simulations sampling subsets of range
stiff = 1     #Stiffness or 'k' value of umbrella potential
```

For the oxDNA simulation parameters, we need to set the number of steps for each window, the temperature, and the intervals at which to print the energy and configuration data. We can also modify other oxDNA parameters as required, such as salt concentration.

```
equilibration_parameters = {'steps':'1e7', 'T':'20C',
                'print_energy_every': '1e6', 'print_conf_interval':'2e6'}
production_parameters = {'steps':'2e7', 'T':'20C',
                 'print_energy_every': '2e6','print_conf_interval':'1e7'}
shared_params = [simulation_manager, n_windows, com_list,
           ref_list, stiff, xmin, xmax]
```

We create an umbrella sampling object and a simulation manager object. If a system directory doesn't exist, instantiating a ComUmbrellaSampling object will create one. If a system directory already exists, it will read the current state of the directory, and will "reattach" to the directory. This means if we need to clear the Jupyter kernel or start a new session with a umbrella simulation run previously, we can "reattach" to the umbrella system directory and preform analysis or continue simulations without modifying the contents or overwriting files. The simulation manager allocates the simulation to the available GPU and CPU resources optimally within on a single node.

```
# Initialize the umbrella sampling and simulation manager objects
us = ComUmbrellaSampling(file_dir, system)
simulation_manager = SimulationManager()
```

Before the simulations are built, we start the NVIDIA multiprocessing service for better performance.

```
# Enable the Nvidia multiprocessing service
simulation_manager.start_nvidia_cuda_mps_control()
```

Now, you'll prepare and run equilibration simulations. These will generate your simulation windows to their assigned sampling range. It is recommended to start with a pre-equilibration check, a brief 50,000 step simulation, to verify your system setup.

If the pre-check is successful, re-run the 'build_equilibration_runs()' method. This will overwrite your previous equilibration setup, emphasis on rebuilding will delete previous files, and start the true equilibration process. The number of steps for this phase depends on your system and order parameter, but for origami-sized systems, a minimum of 2 million steps is suggested.

The 'simulation_manager.run()' method executes these equilibration simulations as a set of 'grandchild' processes. This lets you continue using your notebook while simulations are running. However, if you prefer to block your kernel while running umbrella simulations (necessary for a umbrella sampling python script with Jupyter-validated umbrella setups), use the 'simulation_manager.worker_manager()' method instead. This spawns the simulations as a set of 'child' processes.

```
# Build and run the equilibration simulations
us.build_equlibration_runs(*shared_params, equilibration_parameters)
simulation_manger.run() # This runs the equilibration simulations
```

As the simulations run (or afterwards), you can analyze the time series order parameter values using the following commands:

```
# Analyze the simulation results while they're running or afterwards
us.com_distance_observable(com_list, ref_list)
plt.figure(dpi=200)
for idx in range(0,n_windows,1):
  us.analysis.view_observable('eq', idx, sliding_window=False)
plt.legend(fontsize=4)
```

You can view the final conformation of any window by running the following:

```
us.equilibration_sims[idx].analysis.view_last()
```

Finally, you can proceed to run the production simulation and the Weighted Histogram Analysis Method (WHAM) analysis:

```
us.build_production_runs(*shared_params, production_parameters)
simulation_manger.run()

wham_dir = os.path.abspath('../wham/wham')
n_bins = '200'
tol = '1e-5'
n_boot = '30'
us.wham_run(wham_dir, xmin, xmax, stiff, n_bins, tol, n_boot)
```

Finally, after you have run the WHAM technique, you can view your free energy profile by running:

```
us.plot_free()
```

5 SIMULATION ANALYSIS

After simulations have been completed, it is often necessary to extract the relevant information using statistical analysis. Each case is system-specific, but we provide towards this goal oxDNA Analysis Tools (oat), a Python-based suite designed for analyzing oxDNA simulations. These tools cover some typical use-cases for analysis of simulations, and can also be modified by the users to suite their current needs. They can be used as a series of standalone command-line scripts or as importable Python modules for creating personalized analysis tools. Comprehensive API documentation is available for both top-level analysis and utility APIs at https://lorenzo-rovigatti.github.io/oxDNA/oat/index.html.

6 SIMULATION VISUALIZATION

The viewer oxView.org is a robust, browser-based tool for visualizing and editing oxDNA configurations. Its interface is intuitive, allowing users to simply drag and drop a topology, configuration, or trajectory files into the browser window. It also offers a variety of features, including export to video options, JavaScript console commands, and APIs for scene and edit operations. Additionally, it enables rigid body simulations and 3D printing exports.

Notably, oxView incorporates a "relaxation" simulation, a necessary step prior to the actual data collection to correct potential physical inconsistencies in nucleotide placements. For a more detailed understanding of oxView's capabilities and applications, see (Bohlin et al. 2022).

7 FUTURE DEVELOPMENT

The simulation code is in an active development, and as such future extensions (such as inclusion of more accurate coarse-grained representation of proteins, nanoparticles, support for DNA-RNA hybrids and more accurate representation of different salt conditions, sequence-dependent mechanical properties, capture of hydrodynamic interactions in kinetic simulations) are either being implemented dor on the feature-request lists. Similarly, should new experimental data become available (e.g. thermodynamic measurements of melting temperatures of large DNA nanostructures, kinetic measurements of strand displacement reactions), it will present opportunity for application of novel fitting methods and verification algorithms to better parameterize the model.

The oxDNA and oxRNA simulations can be used to parameterize Markov chain models (e.g. (Zolaktaf et al. 2023)) that describe interactions between DNA and RNA strands in a state space defined by number of base pairs between strands rather than in terms of x-y-z coordinates in 3D space as molecular models do. Where limited experimental data are available, the oxDNA and oxRNA models can be used to augment the training dataset.

ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under Grant DMR-2239518. We acknowledge contributions of Thomas Ouldridge, Flavio Romano, Ben Snodin, Erik Poppleton, and Lorenzo Rovigatti to the oxDNA model and simulation and analysis tools development.

REFERENCES

Bohlin, J., M. Matthies, E. Poppleton, J. Procyk, A. Mallya, H. Yan, and P. Šulc. 2022. "Design and Simulation of DNA, Rna and Hybrid Protein–nucleic Acid Nanostructures With Oxview". *Nature Protocols* 17(8):1762–1788.

Doty, D., B. L. Lee, and T. Stérin. 2020. "Scadnano: a Browser-based, Scriptable Tool for Designing DNA Nanostructures". In *Proceedings of DNA 26: 26th International Conference on DNA Computing and Molecular Programming, September 14-17 2020, Oxford, UK*, 9:1–9:17.

Douglas, S. M., A. H. Marblestone, S. Teerapittayanon, A. Vazquez, G. M. Church, and W. M. Shih. 2009. "Rapid Prototyping of 3d DNA-origami Shapes With Cadnano". *Nucleic Acids Research* 37(15):5001–5006.

Doye, J. P., T. E. Ouldridge, A. a. Louis, F. Romano, P. Šulc, C. Matek, B. E. Snodin, L. Rovigatti, J. S. Schreck, R. M. Harrison et al. 2013. "Coarse-graining DNA for Simulations of DNA Nanotechnology". *Physical Chemistry Chemical Physics* 15(47):20395–20414.

Frenkel, D., and B. Smit. 2001. *Understanding Molecular Simulation: From Algorithms to Applications*. London, UK, Academic Press.

Grossfield, Alan. "wham: the Weighted Histogram Analysis Method". http://membrane.urmc.rochester.edu/wordpress/?page_id=126, accessed on June 15th 2023.

Henrich, O., Y. A. Gutiérrez Fosado, T. Curk, and T. E. Ouldridge. 2018. "Coarse-grained Simulation of DNA Using LAMMPS: an Implementation of the Oxdna Model and Its Applications". *The European Physical Journal E* 41:1–16.

Hong, F., J. S. Schreck, and P. Šulc. 2020. "Understanding DNA Interactions in Crowded Environments With a Coarse-grained Model". *Nucleic Acids Research* 48(19):10726–10738.

Huang, C.-M., A. Kucinic, J. a. Johnson, H.-J. Su, and C. E. Castro. 2021. "Integrated Computer-aided Engineering and Design for DNA Assemblies". *Nature Materials* 20(9):1264–1271.

Lorenz, R., S. H. Bernhart, C. Höner zu Siederdissen, H. Tafer, C. Flamm, P. F. Stadler, and I. L. Hofacker. 2011. "ViennaRNA Package 2.0". *Algorithms for Molecular Biology* 6:1–14.

Maffeo, C., and A. Aksimentiev. 2020. "MrDNA: a Multi-resolution Model for Predicting the Structure and Dynamics of DNA Systems". *Nucleic Acids Research* 48(9):5135–5146.

Mathews, D. H., J. Sabina, M. Zuker, and D. H. Turner. 1999. "Expanded Sequence Dependence of Thermodynamic Parameters Improves Prediction of RNA Secondary Structure". *Journal of molecular biology* 288(5):911–940.

Munsky, B., W. S. Hlavacek, and L. S. Tsimring. 2018. *Quantitative Biology: Theory, Computational Methods, and Models*. Cambridge, USA, MIT Press.

Ouldridge, T. E., A. a. Louis, and J. P. Doye. 2011. "Structural, Mechanical, and Thermodynamic Properties of a Coarse-grained DNA Model". *The Journal of Chemical Physics* 134(8):02B627.

Pinheiro, a. V., D. Han, W. M. Shih, and H. Yan. 2011. "Challenges and Opportunities for Structural DNA Nanotechnology". *Nature Nanotechnology* 6(12):763–772.

Poppleton, E., M. Matthies, D. Mandal, F. Romano, P. Šulc, and L. Rovigatti. 2023. "Oxdna: Coarse-grained Simulations of Nucleic Acids Made Simple". *Journal of Open Source Software* 8(81):4693.

Poppleton, E., R. Romero, A. Mallya, L. Rovigatti, and P. Šulc. 2021. "Oxdna.org: a Public Webserver for Coarse-grained Simulations of DNA and RNA Nanostructures". *Nucleic Acids Research* 49(W1):W491–W498.

Procyk, J., E. Poppleton, and P. Šulc. 2021. "Coarse-grained Nucleic Acid–protein Model for Hybrid Nanotechnology". *Soft Matter* 17(13):3586–3593.

Rothemund, P. W. 2006. "Folding DNA to Create Nanoscale Shapes and Patterns". *Nature* 440(7082):297–302.

Rovigatti, L., J. Russo, F. Romano, M. Matthies, L. Kroc, and P. Šulc. 2022. "A Simple Solution to the Problem of Self-assembling Cubic Diamond Crystals". *Nanoscale* 14(38):14268–14275.

Russo, J., P. Tartaglia, and F. Sciortino. 2009. "Reversible Gels of Patchy Particles: Role of the Valence". *The Journal of Chemical Physics* 131(1):014504.

Sample, Matthew and Matthies, Michael and Šulc, Petr 2023. "Hairygami: Analysis of DNA Nanostructure's Conformational Change Driven By Functionalizable Overhangs". https://arxiv.org/abs/2302.09109, accessed on September 15th 2023.

SantaLucia Jr, J. 1998. "A Unified View of Polymer, Dumbbell, and Oligonucleotide DNA Nearest-neighbor Thermodynamics". *Proceedings of the National Academy of Sciences* 95(4):1460–1465.

Seeman, N. C. 1982. "Nucleic Acid Junctions and Lattices". *Journal of Theoretical Biology* 99(2):237–247.

Seeman, N. C., and H. F. Sleiman. 2017. "DNA Nanotechnology". *Nature Reviews Materials* 3(1):1–23.

Sengar, A., T. E. Ouldridge, O. Henrich, L. Rovigatti, and P. Šulc. 2021. "A Primer On the Oxdna Model of DNA: When to Use It, How to Simulate It and How to Interpret the Results". *Frontiers in Molecular Biosciences* 8:693710.

Snodin, B. E., F. Randisi, M. Mosayebi, P. Šulc, J. S. Schreck, F. Romano, T. E. Ouldridge, R. Tsukanov, E. Nir, A. a. Louis et al. 2015. "Introducing Improved Structural Properties and Salt Dependence into a Coarse-grained Model of DNA". *The Journal of chemical physics* 142(23):06B613_1.

Snodin, B. E., F. Romano, L. Rovigatti, T. E. Ouldridge, A. a. Louis, and J. P. Doye. 2016. "Direct Simulation of the Self-assembly of a Small DNA Origami". *ACS nano* 10(2):1724–1737.

Srinivas, N., T. E. Ouldridge, P. Šulc, J. M. Schaeffer, B. Yurke, A. a. Louis, J. P. Doye, and E. Winfree. 2013. "On the Biophysics and Kinetics of Toehold-mediated DNA Strand Displacement". *Nucleic acids research* 41(22):10641–10658.

Šulc, P., F. Romano, T. E. Ouldridge, J. P. Doye, and A. A. Louis. 2014. "A Nucleotide-level Coarse-grained Model of Rna". *The Journal of chemical physics* 140(23):06B614_1.

Šulc, P., F. Romano, T. E. Ouldridge, L. Rovigatti, J. P. Doye, and A. A. Louis. 2012. "Sequence-dependent Thermodynamics of a Coarse-grained DNA Model". *The Journal of chemical physics* 137(13):135101.

Suma, A., E. Poppleton, M. Matthies, P. Šulc, F. Romano, A. a. Louis, J. P. Doye, C. Micheletti, and L. Rovigatti. 2019. "Tacoxdna: a User-friendly Web Server for Simulations of Complex DNA Structures, From Single Strands to Origami". *Journal of computational chemistry* 40(29):2586–2595.

Tuckerman, M. 2010. *Statistical Mechanics: Theory and Molecular Simulation*. Oxford, UK, Oxford University Press.

Whitelam, S., and P. L. Geissler. 2007. "Avoiding Unphysical Kinetic Traps in Monte Carlo Simulations of Strongly Attractive Particles". *The Journal of chemical physics* 127(15):154101.

Williams, S., K. Lund, C. Lin, P. Wonka, S. Lindsay, and H. Yan. 2009. "Tiamat: a Three-dimensional Editing Tool for Complex DNA Structures". In *Proceedings of DNA 14: 14th International Meeting on DNA Computing, Prague, Czech Republic, June 2-9 2008,*, 90–101.

Zadeh, J. N., C. D. Steenberg, J. S. Bois, B. R. Wolfe, M. B. Pierce, A. R. Khan, R. M. Dirks, and N. A. Pierce. 2011. "Nupack: Analysis and Design of Nucleic Acid Systems". *Journal of Computational Chemistry* 32(1):170–173.

Zolaktaf, S., F. Dannenberg, M. Schmidt, A. Condon, and E. Winfree. 2023. "Predicting DNA Kinetics With a Truncated Continuous-time Markov Chain Method". *Computational Biology and Chemistry*:107837.

AUTHOR BIOGRAPHIES

MICHAEL MATTHIES is a Research Associate and Foresight Fellow. His email address is m.matthies@asu.edu and his homepage is https://www.linkedin.com/in/michael-matthies/. His primary research interest is in developing methods to rationally design nucleic acid nanostructures. To this end he maintains the oxView.org design and viewer tool as well as the oxDNA_analysis_tools simulation analysis package.

MATTHEW L. SAMPLE is a graduate student at the Biodesign Institute at Arizona State University in the Fulton School for Engineering of Matter, Transport, and Energy. His email address is mlsample@asu.edu. His primary research interest consists of applying computational rational design and biomolecular engineering techniques for designing DNA, RNA, and protein based bio-molecular therapeutics.

PETR ŠULC is an Assistant Professor at School of Molecular Sciences at Arizona State University. His primary research interest is in applications of methods of statistical physics and computational physics to studies of DNA and RNA molecules in biology and nanotechnology. His lab co-develops and maintains the oxDNA simulation ecosystem for DNA and RNA nanostructure simulations. His email address is psulc@asu.edu and his homepage is https://sulclab.org.