Parameterizing modified nucleic acids for molecular simulations in the AMBER MD software environment [Article v1.0]

Rodrigo Galindo-Murillo[1], Akanksha Manghrani[2], Daniel Roe[3], Olivia Love[4], Pablo D. Dans[5], Thomas E. Cheatham III[4], Christina Bergonzo[2*]

[1]1Department of Medicinal Chemistry, Ionis Pharmaceuticals, 2855 Gazelle Court, Carlsbad, California 92010, United States; [2]Institute for Bioscience and Biotechnology Research, National Institute of Standards and Technology and the University of Maryland, 9600 Gudelsky Way, Rockville, MD, 20850, United States; [3]Laboratory of Computational Biology, National Heart Lung and Blood Institute, National Institutes of Health, Bethesda, MD 20892, United States; [4]Department of Medicinal Chemistry, College of Pharmacy, University of Utah, 2000 East 30 South Skaggs 306, Salt Lake City, Utah 84112, United States; [5]Computational Biophysics Group, Department of Biological Sciences, CENUR Litoral Norte, Universidad de la República, 50000 Salto, Uruguay. Bioinformatics Unit. Institute Pasteur of Montevideo, Uruguay

This LiveCoMS document is maintained online on GitHub at https://github.com/ManghraniA/ModXNALiveComs; to provide feedback, suggestions, or help improve it, please visit the GitHub repository and participate via the issue tracker.

This version dated August 14, 2025

Abstract Parameterizing modified nucleic acids is a difficult but necessary task for expanding the simulated space of oligonucleotides, including both naturally occurring structures and those with pharmaceutical relevance. In lieu of expensive and difficult chemical synthesis in the laboratory, computer simulations are often performed to make predictions for sequence and structure effects, as well as downstream critical quality attributes. To enable these simulations, modifications have to be parameterized to faithfully represent their effect on nucleotides. This is a non-trivial process, complicated by the fact that it may be the first thing researchers must figure out before they can build their structures and start their initial simulations. To enable these research projects, we created modXNA, a code that assembles pre-parameterized modules of the base, backbone, and sugar, to create bespoke combinations of modifications. In the following tutorial, we provide background on force field parameterization in the Amber software ecosystem and detail the steps necessary to perform parameterization of modified nucleic acids using modXNA.

***For correspondence:**
bergonzoc@umd.edu (CB)

1 Introduction

Molecular dynamics (MD) force fields rely on additive parameters, including bonded and non-bonded terms, to describe a system using physics-based potentials. What follows is a brief overview of Amber force field parameterization, to establish a baseline understanding and provide context for the modXNA [1] parameterization process. The original Amber force field for proteins and nucleic acids addressed parameterization of torsion and angle parameters, deriving these from experimental data [2]. This force field was a precursor to the current equation used to describe the potential energy, introduced in Cornell et al. 1995 [3] (Figure 1). The Cornell et al. Amber ff94 reparametrized the electrostatics using the restrained electrostatic potential (RESP) fitting protocol [3–5]. Subsequent force field modifications undertook refitting nucleic acid sugar pucker and chi torsional parameters [6]. The most recent nucleic acid parameters for RNA combine the refitting of alpha and gamma torsion dihedrals from parmbsc0 [7] with new chi parameters from the OL3 corrections [8], and new chi, epsilon, and zeta torsions in bsc1 [9] for DNA, as well as revisions to beta, alpha, and gamma torsions in the OL15 [9] and OL21 [10] DNA force fields, respectively. In addition to bonded terms, there are non-bonded terms which depend on representative point charges associated with each atom as well as the combination of their atom type specific van der Waals radii. In order to incorporate modifications into the nucleic acid MD, we will have to create parameters for the force field.

$$U = \sum_{\text{bonds}} \boldsymbol{k_r}(r - \boldsymbol{r_0})^2 + \sum_{\text{angles}} \boldsymbol{k_\theta}(\theta - \boldsymbol{\theta_0})^2 + \sum_{\text{dihedrals}} \boldsymbol{k_\phi}[1 + \cos(\boldsymbol{\eta\phi} - \boldsymbol{\gamma_\eta})] + \sum_i \sum_{j \neq i} \epsilon_{i,j} \left[\left(\frac{\boldsymbol{r_{0ij}}}{r_{ij}} \right)^{12} - \left(\frac{\boldsymbol{r_{0ij}}}{r_{ij}} \right)^{6} \right] + \sum_i \sum_{j \neq i} \frac{\boldsymbol{q_i q_j}}{4\pi \epsilon_0 r_{i,j}}$$

Figure 1. AMBER force field equation for molecular dynamics simulations. Bold, blue font denotes variables in the equation that must be parameterized by the user.

When developing parameters for new residues, charges are fit to each atom using RESP fitting [3–5]. RESP fitting [5] begins with calculating the molecular electrostatic potential. The Hartree-Fock (HF) level of theory with the 6-31G* basis set is used due to opportune cancellation of error between the overestimation of the polarity of molecules and popular water models' (TIP3P and SPC) enhanced dipole over gas-phase values. A fixed grid of points is created in the solvent accessible space around the molecule, outside the van der Waals radius of the molecule. Quantum mechanics (QM) is used to calculate the electrostatic potential at each point. Least squares fitting assigns partial charges to each atom in the molecule, with the constraint that the sum of all charges must be equal to the overall charge of the molecule. An added penalty function (the "restrained" part of RESP) eliminates the overestimation of polar atom's calculated electrostatics by scaling down the magnitude of their charges [5].

Nucleic acids were originally RESP fit using a two-stage method, which involved splitting each nucleotide into two representative groups, the deoxyribonucleoside base and dimethylphosphate, representing the backbone phosphate group - only the B-form conformation was considered. In the first stage of fitting, hyperbolic restraints were applied to all heavy atoms. Additional charge constraints were applied to the "joint" regions of each of these functional groups in order to correctly assign charges for the 3' and 5' termini, and to enforce an integer charge for the entire system. The sugar atoms, with the exception of C1' and H1', were intermolecularly equivalenced, meaning that their charges were standardized in all 4 deoxyribonucleosides. The OP1 and OP2 atoms were equivalenced, as well as the hydrogen atoms of the NH2 groups. In the second stage of charge fitting, restraints were increased, freezing heavy atoms and assigning charges for hydrogens of CH3 and CH2 groups. Equivalencing of C2'H2 and C5'H2 groups establishes the same charges for all hydrogens in the group.

2 Scope Of the tutorial

Parameterization of modifications, including the designation of new atom types with their corresponding hybridization, must be developed for use in MD simulations. Based on the current form of the force field (Figure 1), we need to develop parameters for bonds, angles, dihedrals, and non-bonded interactions, including van der Waals and electrostatics.

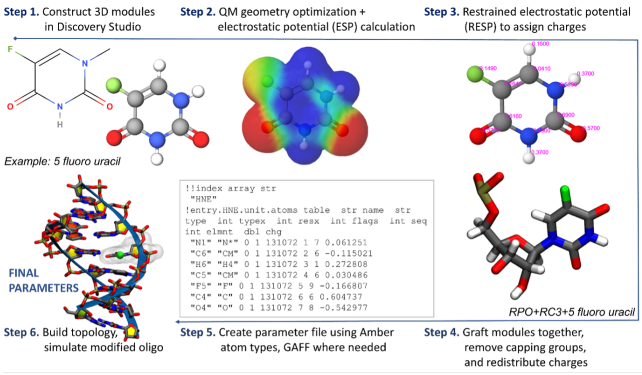

Figure 2. A schematic of the workflow used in modXNA.

The updated procedure for parameterization, introduced in modXNA [1], preserves some of these early features – specifically, we wanted to make sure that in the absence of a modification, the underlying nucleotide parameters are identical to the current force field, that intermolecular

equivalences were preserved, and that joint regions were treated fairly (Figure 2). The split from two components (a base containing a deoxyribose and a phosphate group) to three components (base, sugar, and backbone) was undertaken in order to modify the smallest possible unit (Figure 3). Three modules are chosen by the user: one backbone, one sugar, and one base. The Catalog hosted online (at https://modxna.chpc.utah.edu) and on the GitHub repository for this publication outlines the fragments that have already been parameterized. The three-letter "Fragment IDs" for each fragment are designated in the input file in order: [backbone] [sugar] [base].

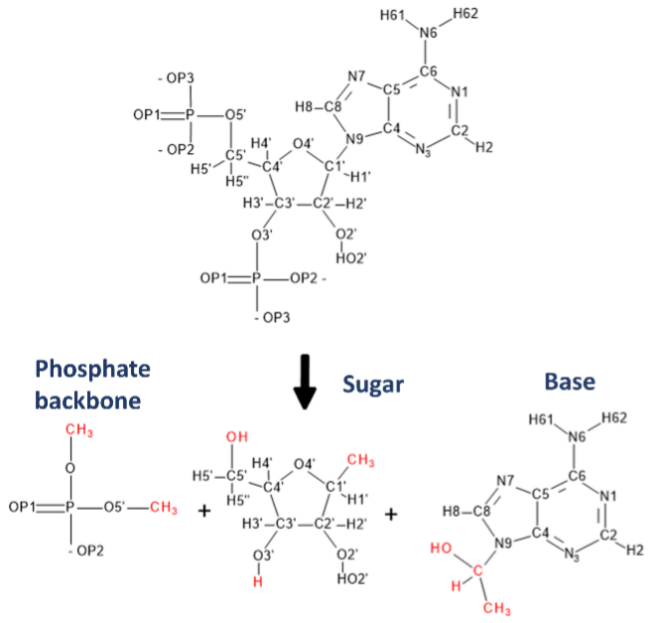

Figure 3. Modules which make up the final nucleotide. All three parts are required for modXNA. The red groups denote "capping groups" that were present during QM calculations to act as placeholders for the other modules that make up a complete nucleotide. These capping groups are removed when modXNA assembles the residue.

2.1 Using modXNA.sh, prerequisites

After choosing the three fragments, the user runs the modXNA.sh script, which is distributed as an accessory program within AmberTools25. The script utilizes Amber's LEaP (tleap), CPPTRAJ [11] and sander programs. Header lines in each fragment mol2 file are used to set flags which strip capping groups, allowing the combination of fragments into a residue. After joint corrections are applied, and charge equivalencing occurs, the full nucleotide residue is built in tleap, and quickly minimized using sander. A final step sees the nucleotide residue loaded into tleap to create an Amber "lib" (library) file. The final output of the program, necessary to build this modified residue in tleap, is the XXX.lib file, where "XXX" is the random 3 letter name set as this specific modification's residue name. Additional temporary outputs include topology, coordinates, and mol2 or pdb files from each step of the stitching together of the modules, which is useful to ensure the correct modification is being added to the correct location.

In addition to charge assignment based on atom types in each module, the other parameters of the nucleic acid force field must be assigned. Bonds, angles, and dihedrals are assigned based on similarity to current parameters or through the General Amber Force Field (GAFF) [12]. Van der Waals radii are similarly assigned through comparison.

3 Building modified oligonucleotides with modXNA

3.1 Incorporating pseudo uridine in an RNA duplex

A simple initial example is running MD simulations of an RNA duplex that contains a pseudo-uridine (PSU) modification. We will base this example on the Nuclear Magnetic Resonance (NMR) determined structure deposited with PDB ID 6I1W [13] that contains the PSU residue at position 5 of the sense strand. After downloading the legacy PDB Format file (6I1W.pdb) some cleaning is required to have a working PDB file that we can then use for building parameter and input coordinate files for MD simulations. There are multiple ways to do this; in this case, we will use the prepareforleap command available in CPPTRAJ [14]. Briefly, this command removes information that the PDB file includes that is either not needed by or would cause issues with LEaP, removes water molecules, and performs a brief check of the residues to report problems that LEaP might encounter. In a text file (prepareforleap.cpptraj), add the commands (Listing 1):

Listing 1. prepareforleap.cpptraj

```
parm 6I1W.pdb
loadcrd 6I1W.pdb name edit
prepareforleap crdset edit name FromPrepForLeap \
      out FromPrepForLeap-tleap.in leapunitname x  \
      pdbout FromPrepForLeap.pdb nowat noh
go
```

we execute the command using:

```
$ cpptraj -i prepareforleap.cpptraj
```

CPPTRAJ will open the 6IW.pdb file, delete the water molecules, strip hydrogens and generate a new PDB file (from-prepareforleap.pdb). The CPPTRAJ output will print a warning referring to the PSU residue:

```
Potential problem: PSU_5_A is an
unrecognized name and may not have
parameters.
```

This is because PSU is not a residue with parameters in the standard AMBER force fields. We will now use modXNA to create the parameters for the PSU residue. Even though modXNA is available in Ambertools25, it is recommended that the user download the latest version of modXNA from the repository (https://modxna.chpc.utah.edu/ or https://github.com/cbergonzo/LiveCommsModXNATutorial). As commented in the introduction, modXNA requires an input file that declares the backbone, the sugar, and the base. Looking at the catalog of available fragments on the modXNA website, we need the RNA phosphate backbone (modXNA code RPO), a ribose sugar (modXNA code RC3), and the pseudo uridine base (modXNA code PUU). The input file (in.modxna, Listing 2) is:

Listing 2. in.modxna

```
### modXNA input file
RPO RC3 PUU
```

We run the script with:

```
$ modXNA.sh -i in.modxna -m PSU
```

The -i flag indicates the input file, and the -m flag indicates the name that we want for the generated nucleotide. Without the -m flag, modXNA will generate a random 3-letter residue name. The script will generate multiple temporary files (with the prefix "tmp.") that are useful to troubleshoot if something went wrong. If the script is completed successfully, we will have a PSU.lib file with the nucleotide, created from the selected fragments in the modXNA input script. We can check the structure of the generated nucleotide using the tmp.opt.pdb file in any molecular visualizer (Figure 4).

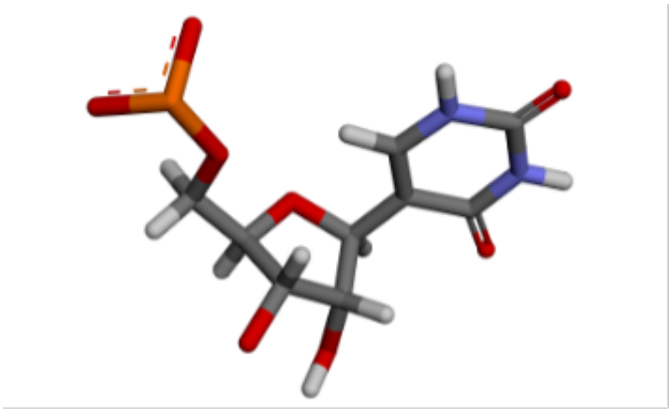

Figure 4. Visualization of the tmp.opt.pdb structure for new residue name PSU.

We can now build our Amber topology and coordinates files using LEaP. We use the following tleap script (build.tleap, Listing 3):

Listing 3. build.tleap

```
##########################
source leaprc.DNA.OL15
source leaprc.RNA.OL3
source leaprc.gaff2
source leaprc.water.opc
##########################
## Load the LIB file created with modXNA
loadoff PSU.lib
## Load the force mod file included in modXNA
loadamberparams /home/user/modXNA/dat/frcmod.modxna
#####################################
x = loadpdb FromPrepForLeap.pdb
solvateoct x OPCBOX 9.0
addionsrand x Na+ 0
addionsrand x Cl- 0
saveamberparm x complex.parm7 complex.crd
charge x
quit
```

We can run the above script with the following command:

```
$ $ tleap -s -f build.tleap
```

LEaP will load the declared AMBER force fields, load the LIB file and the frcmod files, then load the PDB structure generated from the prepareforleap command. The output will print several warnings:

```
FATAL:  Atom .R<PSU 5>.A<N1 31> does not
have a type.
FATAL:  Atom .R<PSU 5>.A<C5 32> does not
have a type.
```

This means that the atom names found in the FromPrepareForLeap.pdb structure do not match the LIB file we generated with modXNA, due to have different names. For this specific example, we can safely delete the lines that contain the atoms N1 and C5 of the PSU residue from the FromPrepareForLeap.pdb file. LEaP will fill in the missing atoms following the PSU.lib library file. We can edit the FromPrepareForLeap.pdb file and delete lines for Atom 83 and 87 as shown in Listing 4. Now re-run the command:

```
tleap -s -f build.tleap
```

The files complex.parm7 and complex.crd will be successfully generated.

3.2 Incorporating 2'-O-methylated (O-methyl) nucleotides in an RNA duplex

The second example will consist of a DNA-RNA hybrid where the DNA is incorporated by 2'-O-methyl ribose sugar

Listing 4. PSU PDB file

```
ATOM      84  C2  PSU A   5       1.517   1.054  13.561  1.00  0.00           C
ATOM      85  N3  PSU A   5       0.817   1.039  14.737  1.00  0.00           N
ATOM      86  C4  PSU A   5       1.344   1.223  15.994  1.00  0.00           C
ATOM      88  C6  PSU A   5       3.502   1.530  14.890  1.00  0.00           C
ATOM      89  O2  PSU A   5       0.966   0.847  12.486  1.00  0.00           O
ATOM      90  O4  PSU A   5       0.588   1.145  16.960  1.00  0.00           O
ATOM      91  C1' PSU A   5       3.445   1.699  17.400  1.00  0.00           C
ATOM      92  C2' PSU A   5       3.572   0.372  18.161  1.00  0.00           C
ATOM      93  O2' PSU A   5       3.461   0.635  19.562  1.00  0.00           O
ATOM      94  C3' PSU A   5       4.979  -0.073  17.742  1.00  0.00           C
ATOM      95  C4' PSU A   5       5.696   1.274  17.808  1.00  0.00           C
ATOM      96  O3' PSU A   5       5.531  -1.048  18.615  1.00  0.00           O
ATOM      97  O4' PSU A   5       4.770   2.215  17.268  1.00  0.00           O
ATOM      98  C5' PSU A   5       7.047   1.295  17.081  1.00  0.00           C
ATOM      99  O5' PSU A   5       6.908   1.097  15.683  1.00  0.00           O
ATOM     100  P   PSU A   5       8.172   1.196  14.691  1.00  0.00           P
ATOM     101  OP1 PSU A   5       9.338   0.548  15.339  1.00  0.00           O
ATOM     102  OP2 PSU A   5       7.743   0.714  13.358  1.00  0.00           O
```

nucleotides in the anti-sense strand. The PDB 1NAO [16] will serve as a template. As before, we download the PDB file and use a similar CPPTRAJ script to obtain a pruned 'clean' PDB (prepareforleap.cpptraj, Listing 5):

Listing 5. prepareforleap.cpptraj

```
parm 1nao.pdb
loadcrd 1nao.pdb name edit
prepareforleap crdset edit name FromPrepForLeap \
      out FromPrepForLeap-tleap.in leapunitname x  \
      pdbout FromPrepForLeap.pdb nowat noh
go
```

Run the script with:

```
$ cpptraj -i prepareforleap.cpptraj
```

and we get several warnings:

```
Potential problem: OMG_10_B is an
unrecognized name and may not have
parameters.
Potential problem: OMU_11_B is an
unrecognized name and may not have
parameters.
Potential problem: OMC_12_B is an
unrecognized name and may not have
parameters.
Potential problem: OMC_17_B is an
unrecognized name and may not have
parameters.
Potential problem: OMC_18_B is an
unrecognized name and may not have
parameters.
```

These are the C2'-O-methyl residues that are not included in the AMBER force fields for standard nucleotides. We will use modXNA to create the following new residues, outlined in Table 1: 5MG, OMU, OMC, 3MC.

Table 1. Description of input file modules for building four modified residues in PDBID 1NAO.

Residue name	Backbone fragment code	Sugar fragment code	Base fragment code	Notes
5MG	5PO	OME	DGG	5'-terminal
OMU	DPO	OME	RUU	
OMC	DPO	OME	DCC	
3MC	DPO	OME	DCC	3'-terminal

It is important to note that modXNA does not create 5' or 3' terminal nucleotides, only central nucleic acid fragments that are expected to have a 5' and 3' bond. For this system, we require a 5'-terminal modified residue (guanine with C2'-O-methyl sugar) and a 3'-terminal modified residue (cytosine with C2'-O-methyl sugar). We will first focus on the central residues: a C2'-O-methyl uridine and a C2'-O-methyl cytosine are required to match the sequence from the experimental structure.

Create a new directory with the name OMU. Inside the directory, create the modXNA input file (OMU.in.modxna, Listing 6):

Listing 6. OMU.in.modxna

```
### C2'-OMe-uridine
DPO OME RUU
```

Then, run modXNA.sh with:

```
$ modXNA.sh -i in.modxna -m OMU
```

Create a new directory with the name OMC. Inside the directory, create the modXNA input file (OMC.in.modxna, Listing 7):

Listing 7. OMC.in.modxna

```
### C2'-OMe-cytosine
DPO OME DCC
```

Run the modXNA.sh script inside the OMC directory:

```
$ modXNA.sh -i in.modxna -m OMC
```

We can open the files OMU/tmp.opt.pdb and OMC/tmp.opt.pdb and confirm that both nucleotides have been created correctly (Figure 5).

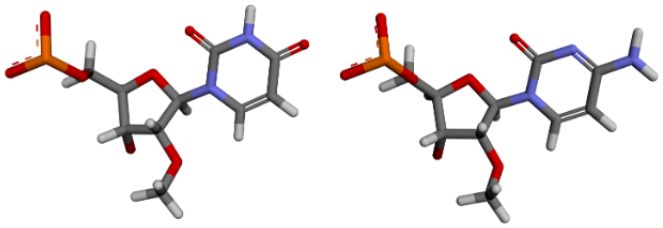

Figure 5. Left: OMU, right: OMC. Both structures are named tmp.opt.pdb, created using modXNA

These residues are positions OMU11, OMC12 and OMC17 of the anti-sense strand (B chain) of the original 1NAO structure. To generate the 5'-terminal OMG residue, we can use the same modXNA script with the option `--5cap`. Currently, this option only works with the backbone fragment 5PO, which has a capped 5'-side, otherwise we would have missing atoms in the generated nucleotide. We are going to name the 5'-terminal residue 5MG, to not confuse with a central fragment. Create a new folder with the name 5MG, as well as the modXNA input file (5MG.in.modxna, Listing 8):

Listing 8. 5MG.in.modxna

```
### C2'-OMe-guanine
5PO OME DGG
```

Run the modXNA.sh script inside the OMC directory:

```
$ modXNA.sh -i in.modxna -m 5MG --5cap
```

Notice the extra flag `--5cap` to indicate to modXNA that we want a 5'-terminal residue. For the 3'-terminal residue, create a new folder with the name 3MC and create the modXNA input file (3MC.in.modxna, Listing 9):

Listing 9. 3MC.in.modxna

```
### C2'-OMe-cytosine
DPO OME DCC
```

And run modXNA:

```
$ modXNA.sh -i in.modxna -m 3MC --3cap
```

Notice the extra flag `--3cap` to indicate to modXNA that we want a 3'-terminal residue. The resulting 5'-terminal and 3'-terminal are shown in Figure 6. We have now all the required library files (5MG.lib, OMU.lib, OMC.lib and 3MC.lib) to build the RNA-DNA hybrid that matches the structure.

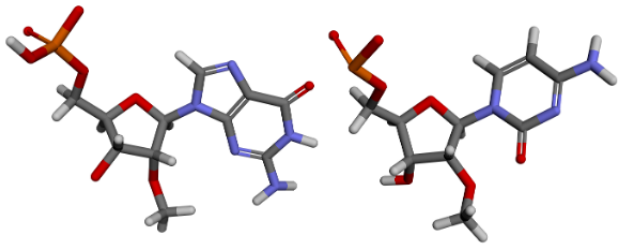

Figure 6. Left: 5' terminal 5MG residue. Right: 3' terminal 3MC residue. Both are the tmp.opt.pdb files created by modXNA.

3.3 Building a new modified sequence

In this next example, we will use modXNA to build a single strand antisense oligonucleotide (ASO) that complements with an RNA target. The visual representation of the ASO is depicted in Figure 7.

Figure 7. Sketch representation of the ASO. Sugar modifications: Blue is cET (k), red is C2'-O-methyl (m) and orange is C2'-O-methoxyethyl (e). Backbone modifications: red circle between the bases is phosphate linkage and grey circle with an 's' is Phosphorothioate linkage. The ^{m}C is 5 methyl cytosine.

ASOs are typically heavily modified and contain a hybrid DNA-RNA-DNA strand. In the example here, the modifications include a 5 methyl cytosine base (M5C) modification, a gapmer (CET) sugar modification, a 2'-O-methoxyethyl (MOE) sugar modification, a 2'-O-methyl (OME) sugar modification, and a phosphorothioate (PS1) backbone modification. To build the ASO, we need the above modifications combined into the following new nucleotides: 5CG, CEG, MEC, MOG, MOA, POG, POA, MSC, OMU, 3EC, detailed in Table 2.

All the new residues follow the same procedure:

1. Make a new folder (to keep things organized) with the new residue name (i. e. 5CG)

Table 2. Description of input file modules for building four modified residues in model ASO.

Residue name	Backbone fragment code	Sugar fragment code	Base fragment code	Details of fragments	Notes
5CG	5PO	CET	DGG	Phosphate backbone + cET + guanine	5'-terminal
CEG	DPO	CET	DGG	Phosphate backbone + cET + guanine	Central residue
MEC	DPO	CET	M5C	Phosphate backbone + cET + 5 methyl cytosine	Central residue
MOG	DPO	MOE	RGG	Phosphate backbone + MOE + guanine	Central residue
MOA	DPO	MOE	RAA	Phosphate backbone + MOE + adenine	Central residue
POG	PS1	MOE	RGG	Phosphorothioate (S) backbone + MOE + guanine	Central residue
POA	DPO	MOE	RAA	Phosphorothioate (S) backbone + MOE + adenine	Central residue
MSC	PS1	CET	M5C	Phosphorothioate (S) backbone + cET + 5 methyl cytosine	Central residue
OMU	PS1	OME	RUU	Phosphorothioate (S) backbone + O-methyl + uracil	Central residue
3EC	PS1	CET	M5C	Phosphorothioate (S) backbone + cET + 5 methyl cytosine	3'-terminal

2. In the new folder, make a modXNA input file with the fragment code of the backbone, sugar and base (in.modxna with the codes: 5PO CET DGG, for the residue 5CG)
3. Run modXNA (remember to use the `--5cap` for the 5CG residue and `--3cap` flag for the 3EC residue):

```
$ modXNA.sh -i in.modxna -m 5CG --5cap
```

4. Check that the new residues have been built successfully by opening the tmp.opt.pdb structure.

The resulting structures are shown in Figure 8. Before we can use these 10 library files, we need to combine all the different LIB files into a single file. This will make it easier to load just one bigger LIB file into LEaP instead of loading 10 independent LIB files. To do this, create the text file (create-ASO-lib.tleap, Listing 10):

Listing 10. create-ASO-lib.tleap

```
loadoff 3EC/3EC.lib
saveoff 3EC ASO.lib
loadoff 5CG/5CG.lib
saveoff 5CG ASO.lib
loadoff CEG/CEG.lib
saveoff CEG ASO.lib
loadoff MEC/MEC.lib
saveoff MEC ASO.lib
loadoff MOA/MOA.lib
saveoff MOA ASO.lib
loadoff MOG/MOG.lib
saveoff MOG ASO.lib
loadoff MSC/MSC.lib
saveoff MSC ASO.lib
loadoff OMU/OMU.lib
saveoff OMU ASO.lib
loadoff POA/POA.lib
saveoff POA ASO.lib
loadoff POG/POG.lib
saveoff POG ASO.lib
quit
```

Run the file with:

```
$ tleap -s -f create-ASO-lib.tleap
```

This will load all the LIB files we created with modXNA and save the information into the same 'ASO.lib' file. If you inspect the ASO.lib file generated by tleap, you will see that all the new residues are in one single file.

Now that we have all the required nucleotides, we need to build the ASO. There are several methodologies that we

can use to proceed. The simplest one is generating a single strand with the correct sequence of nucleotide analogs using the 'sequence' command available in LEaP, using the following script (Aso-single-strand.tleap, Listing 11):

Listing 11. Aso-single-strand.tleap

```
############################
source leaprc.DNA.OL24
source leaprc.RNA.OL3
source leaprc.gaff2
source leaprc.water.opc
############################
loadoff ASO.lib
loadamberparams /home/user/modXNA/dat/frcmod.modxna
############################
ASO=sequence{5CG CEG MEC MOG MOA MOG POG POA MSC OMU
 3EC}
solvateoct ASO OPCBOX 9.0
addionsrand ASO Na+ 0
addionsrand ASO Cl- 0
saveamberparm ASO ASO-ss.parm7 ASO-ss.crd
quit
```

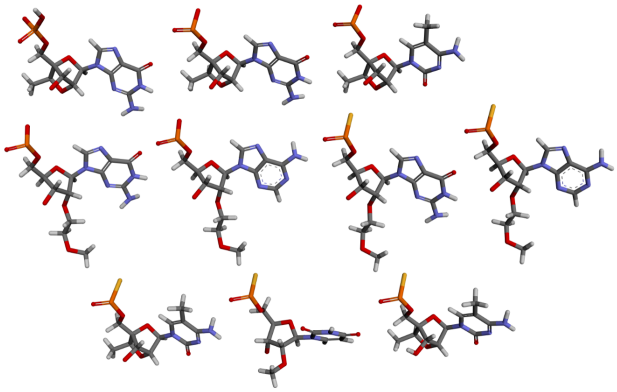

Figure 8. Visualizations of all the tmp.opt.pdb files generated by modXNA for each of the new nucleotides.

LEaP will create a linearized structure with the correct sequence of residues. It will have some structural overlaps, however, and if we run a quick structural minimization using AMBER/sander (using the 9-step protocol from AmberMDPrep [15]), we obtain a structure with no clashes, shown in Figure 9.

4 Conclusions

In the above tutorial, we have outlined the use of modXNA.sh, a script and database of pre-parameterized modules, in order to create bespoke modified nucleotides. We have shown examples of building structures pulled from the PDB, which contain modified residues (1NAO and 6I1W), as well as using the program to create a heavily modified single stranded RNA ASO. We have outlined the procedure as well as identified some common pitfalls. While we know that building new residues for MD simulations can be difficult, we hope this work streamlines the process and decreases the time-to-simulating for users.

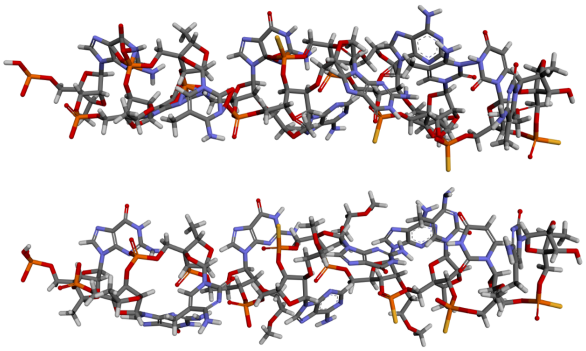

Figure 9. Top: PDB structure generated directly from the ASO-single-strand.parm7 and ASO-single-strand.crd. Bottom: structure obtained after running minimization/equilibration using AmberMDPrep.

5 Content and links

All scripts used in this tutorial as well as the modXNA script are available at:
https://github.com/ManghraniA/ModXNALiveComs

Author Contributions

CB and RGM wrote the manuscript; RGM performed simulations; DRR, PDD, TEC3, OL, and AM contributed to proofreading and testing; AM formatted for publication.

Potentially Conflicting Interests

The authors declare no competing financial interest.

Funding Information

The authors gratefully acknowledge the computing and data resources from the Center for High-Performance Computing at the University of Utah and the University of Maryland at IBBR. Funding for this research came from the National Institutes of Health R-01 GM-081411. D.R.R. was funded through NHLBI. P.D.D. was funded through CSIC I+D 22520220100376UD.

Disclaimer

identification does not imply recommendation or endorsement by the National Institute of Standards and Technology, nor does it imply that the materials or equipment identified are necessarily the best available for the purpose.

Author Information

ORCID:

Rodrigo Galindo-Murillo: orcid.org/0000-0001-5847-4143
Akanksha Manghrani: orcid.org/0000-0002-6498-7305
Daniel R. Roe: orcid.org/0000-0002-5834-2447
Olivia Love: orcid.org/0000-0002-7743-8495
Pablo D. Dans: orcid.org/0000-0002-5927-372X
Thomas E. Cheatham III: orcid.org/0000-0003-0298-3904
ChristinaBergonzo: orcid.org/0000-0003-1990-2912

References

[1] **Love O**, Galindo-Murillo R, Roe DR, Dans PD, Cheatham TEI, Bergonzo C. modXNA: A Modular Approach to Parametrization of Modified Nucleic Acids for Use with Amber Force Fields. Journal of Chemical Theory and Computation. 2024; 20:9354–9363. https://doi.org/10.1021/acs.jctc.4c00767.

[2] **Weiner SJ**, Kollman PA, Nguyen DT, Case DA. An all atom force field for simulations of proteins and nucleic acids. Journal of Computational Chemistry. 1986; 7(2):230–252. https://doi.org/10.1002/jcc.540070216.

[3] **Cornell WD**, Cieplak P, Bayly CI, Kollman PA. Application of RESP Charges To Calculate Conformational Energies, Hydrogen Bond Energies, and Free Energies of Solvation. Journal of the American Chemical Society. 1993; 115:9620–9631. https://doi.org/10.1021/ja00074a030.

[4] **Cieplak P**, Cornell WD, Bayly C, Kollman PA. Application of the multimolecule and multiconformational RESP methodology to biopolymers: Charge derivation for DNA, RNA, and proteins. Journal of Computational Chemistry. 1995; 16:1357–1377. https://doi.org/10.1002/jcc.540160106.

[5] **Bayly CI**, Cieplak P, Cornell WD, Kollman PA. A well-behaved electrostatic potential based method using charge restraints for deriving atomic charges: The RESP model. Journal of Physical Chemistry. 1993; 97:10269–10280. https://doi.org/10.1021/j100142a004.

[6] **Cheatham TEI**, Cieplak P, Kollman PA. A modified version of the Cornell et al. force field with improved sugar pucker phases and helical repeat. Journal of Biomolecular Structure and Dynamics. 1999; 16:845–862. https://doi.org/10.1080/07391102.1999.10508297.

[7] **P'erez A**, March'an I, Svozil D, Sponer J, Cheatham TEI, Laughton CA, Orozco M. Refinement of the AMBER force field for nucleic acids: Improving the description of α/γ conformers. Biophysical Journal. 2007; 92:3817–3829. https://doi.org/10.1529/biophysj.106.097782.

[8] **Zgarbov'a M**, Otyepka M, Sponer J, Mladek A, Banas P, Cheatham TEI, Jurecka P. Refinement of the Cornell et al. nucleic acids force field based on reference quantum chemical calculations of glycosidic torsion profiles. Journal of Chemical Theory and Computation. 2011; 7:2886–2902. https://doi.org/10.1021/ct200162x.

[9] **Ivani I**, Dans PD, Noy A, Perez A, Faustino I, Hospital A, Walther J, Andrio P, Goni R, Balaceanu A, Portella G, Battistini F, Gelpi JL, Gonzalez C, Vendruscolo M, Laughton CA, Harris SA, Case DA, Orozco M. Parmbsc1: A refined force field for DNA simulations. Nature Methods. 2015; 13:55–58. https://doi.org/10.1038/nmeth.3658.

[10] **Zgarbov'a M**, Sponer J, Jurecka P. Z-DNA as a Touchstone for Additive Empirical Force Fields and a Refinement of the Alpha/Gamma DNA Torsions for AMBER. Journal of Chemical Theory and Computation. 2021; 17:6292–6301. https://doi.org/10.1021/acs.jctc.1c00755.

[11] **Roe DR**, Cheatham TEI. PTRAJ and CPPTRAJ: Software for processing and analysis of molecular dynamics trajectory data. Journal of Chemical Theory and Computation. 2013; 9(7):3084–3095. https://doi.org/10.1021/ct400341p.

[12] **Wang J**, Wolf RM, Caldwell JW, Kollman PA, Case DA. Development and testing of a general Amber force field. Journal of Computational Chemistry. 2004; 25:1157–1174. https://doi.org/10.1002/jcc.20035.

[13] **Deb I**, Popenda L, Sarzynska J, Lahiri A, Walczak A, Biala E, Gdaniec Z, Kierzek R, Banerjee R. Computational and NMR studies of RNA duplexes with an internal pseudouridine-adenosine base pair. Scientific Reports. 2019; 9. https://doi.org/10.1038/s41598-019-55191-4.

[14] **Roe DR**, Bergonzo C. prepareforleap: An automated tool for fast PDB-to-parameter generation. Journal of Computational Chemistry. 2022; 43(13):930–935. https://doi.org/10.1002/jcc.26804.

[15] **Roe DR**, Brooks BR. A protocol for preparing explicitly solvated systems for stable molecular dynamics simulations. Journal of Chemical Physics. 2020; 153(5):054123. https://doi.org/10.1063/5.0014153.