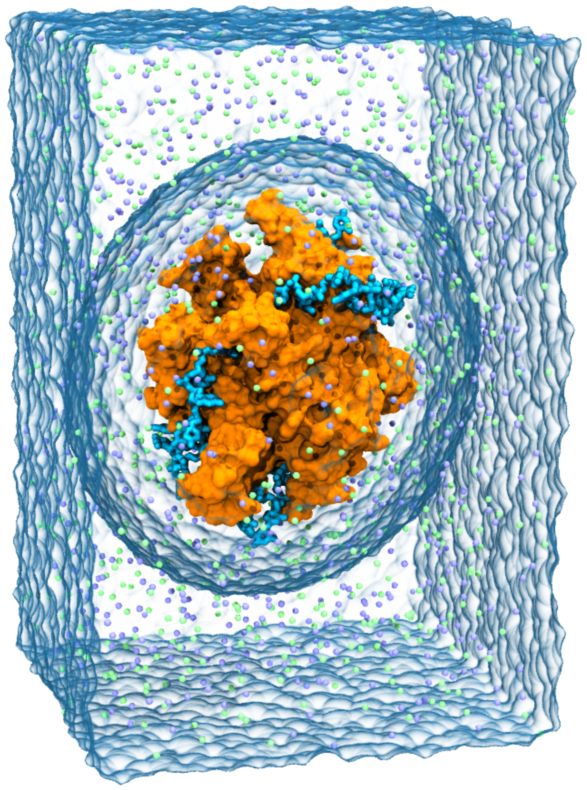
**Aksimentiev Group**
**Department of Physics and**
**Beckman Institute for Advanced Science and Technology**
**University of Illinois at Urbana-Champaign**

# Introduction to MD simulation of DNA–protein systems



**Chris Maffeo**
**Rogan Carr**
**Aleksei Aksimentiev**

## Contents

## 1 Introduction

DNA is so famously known as the carrier of genetic information that the structural and dynamical aspects of the molecule are often neglected. However, most cellular processes that involve DNA cannot be understood without taking into account its physical properties and structure.

Single-stranded DNA (ssDNA) is a polymer composed of nucleotides. A nucleotide consists of one of four hydrophobic, ring-shaped bases (A, T, C or G) connected to a sugar ring, which is in turn bonded to a phosphate. The phosphate of one nucleotide can be connected to the sugar ring of another. When this process is repeated, ssDNA is formed.

If two strands have complementary sequences (A·T or C·G), they can anneal to form a double-stranded DNA (dsDNA) duplex stabilized by base-stacking and Watson-Crick hydrogen-bonding between the complementary bases. Canonical DNA (B-DNA) in electrolyte forms a right-handed double-helix. Traversing the duplex by one basepair corresponds to a rotation of about $34°$. A DNA duplex—the smallest self-assembled unit of DNA—is used by the cell for packaging and protecting its genetic information. DNA-DNA and DNA-protein interactions can give rise to self-assembled structures; the DNA double-helix wraps twice around a histone to form the nucleosome, which in turn form aggregates that eventually form chromatin—the fiber that makes up the chromosome [1].

During DNA replication, the cell's machinery unravels these structures, forking dsDNA into a pair of single DNA strands at the last step. A protein called DNA polymerase moves along each unwound ssDNA strand to synthesize a complementary strand. After the DNA is unwound, but before the DNA polymerase arrives, single-stranded DNA binding protein (SSB) wraps up the ssDNA to prevent the strands from annealing, protect the nucleobases from chemical modifications and prevent the formation of hairpin structures in repetitive, self-complementary regions of DNA [2, 5].

$$\begin{aligned}
U_{\text{total}} &= \sum_{\text{bonds } i} k_i^{\text{bond}} \left(r_i - r_{0i}\right)^2 + \sum_{\text{angle } i} k_i^{\text{angle}} \left(\theta_i - \theta_{0i}\right)^2 \\
&\quad + \sum_{\text{dihedrals } i} k_i^{\text{dihed}} \begin{cases} \left[1 + \cos\left(n_i \phi_i - \gamma_i\right)\right] & n_i \neq 0 \\ \left(\phi_i - \gamma_i\right)^2 & n_i = 0 \end{cases} \\
&\quad + \sum_i \sum_{j>i} 4\epsilon_{ij} \left[ \left(\frac{\sigma_{ij}}{r_{ij}}\right)^{12} - \left(\frac{\sigma_{ij}}{r_{ij}}\right)^{6} \right] + \sum_i \sum_{j>i} \frac{q_i q_j}{4\pi\epsilon r_{ij}}
\end{aligned}$$

Figure 1: The MD potential, where $\mathbf{F} = -\boldsymbol{\nabla} U$.

Small molecules that drive cellular processes can be studied using a variety of techniques. In the Chemla lab, optical traps are used to apply and measure forces acting on single molecules. In the Aksimentiev group, models of single molecules can be manipulated in similar ways. We can apply and measure forces with a computational technique called Molecular Dynamics (MD) simulation.

In MD simulations, molecules are treated as collections of point particles which interact via a set of forces; Newton's equation ($\mathbf{F} = m\mathbf{a}$) is integrated to describe the temporal evolution of the system. MD simulations use a force field, which is a set of equations and parameters that together determine how any pair of point particles interact. The most popular force fields for MD simulation describe biomolecules as collections of atoms which are connected by harmonic bonds (two-body interactions), angles (three-body interactions) and dihedrals (four-body interactions) and interact through the Coulomb and van der Waals potentials. Given the positions and velocities for all the atoms in a system, NAMD (the MD package that we will be using) calculates new positions and velocities using the force on each atom with the equation in Figure 1.

Today, you will prepare a system for a steered molecular dynamics (SMD) simulation of ssDNA and SSB, running briefly to ensure that everything worked. Unfortunately, there is insufficient time to perform a long-timescale simulation, so a final trajectories of an equivalent simulation is provided. You will then perform simple analysis of the trajectory using VMD's tcl interface.

This guide is a modified version of a complete tutorial, **Introduction to MD simulation of DNA-protein systems**, which covers the basics of system assembly and simulation with NAMD more extensively. The complete tutorial introduces the reader to the MD method slowly and pays careful attention to details This guide rushes through many of these details, so please ask if you have questions!

## 2 System setup and simulation in general

In order to perform an MD simulations using NAMD, you must have at least three files:

1. a PDB containing information about the coordinates and names of each atom;
2. a PSF containing information that will later be used by NAMD to decide what forces to apply to each atom, including the mass, charge, and atom connectivity (bonds, angles, dihedrals and impropers), as well as the atom type, which specifies van der Waals radius and depth; and
3. a NAMD configuration file that instructs NAMD what and how to run the simulation.

SSB is known to bind ssDNA in two modes that depend on ion concentration: $SSB_{35}$ and $SSB_{65}$ [4]. The subscript denotes the approximate number of nucleotides occluded in each mode. $SSB_{35}$ binds ssDNA cooperatively, and can form indefinitely long protein clusters; $SSB_{65}$ binds ssDNA with limited cooperativity. Both binding modes are believed to have functional roles in the cell.

An x-ray structure containing two $\sim 30$ basepair ssDNA fragments bound to SSB was recently published [4]. The structure depicts a homotetramer with folds that accommodate the DNA, which is held in place through a mix of base-stacking and electrostatic interactions. Models of $SSB_{35}$ and $SSB_{65}$, made by extending the DNA fragments in the crystal structure, are shown in Fig. 2a and b. Here we focus on $SSB_{65}$, which is more prevalent at physiological ion concentrations.

You will study SSB by pulling on DNA bound to SSB to force its dissociation. To be computationally economical, the DNA was pulled along an unusual axis so that the DNA fit between periodic images of SSB when fully stretched. This axis was chosen by trial and error, rotating the extended DNA and adjusting the size of the unit cell until a suitable pathway was obtained. There may be more thoughtful ways of picking an axis, but this is a one-time task and we chose a quick, guess-and-check approach. This approach can be useful, but it is generally better to use a simulation system that is large enough to accommodate the DNA or to periodically truncate the excess DNA.

## 3 System assembly

A project typically begins with extensive review of the literature about the system, in this case SSB (PDB accession code: 1EYG). Particular attention

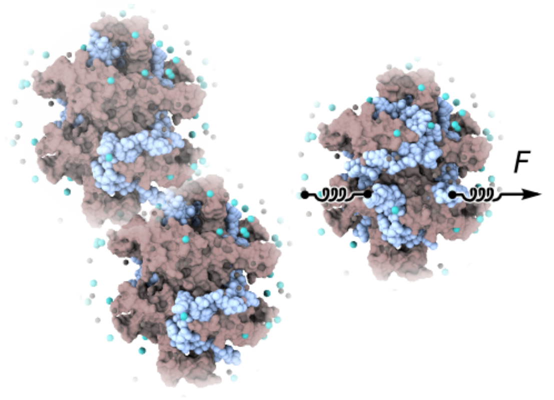


Figure 2: Models of ssDNA bound to SSB. (a) The ends of ssDNA bound to $SSB_{35}$ extend from opposite sides of SSB, allowing unlimited cooperativity. (b) The ends of the ssDNA bound to $SSB_{65}$ extend on the same side of SSB, allowing only limited cooperativity. A method proposed to remove DNA from SSB is illustrated with cartoon springs. The DNA is represented as light-blue van der Waals spheres; the surface of SSB is shown in pink; $K^+$ and $Cl^-$ are shown as small brown and cyan spheres, respectively.

should be paid to any crystallographic articles reporting structures that will provide the initial configuration of the system. It is good to carefully examine the structure to obtain as complete an understanding of the protein as possible before simulating; many trivial errors can be avoided early by doing so. For example, protein structures may be missing residues (several missing residues in the 1EYG structure were added through homology modeling before producing `ssb65.pdb`). More subtly, a number of chemical modifications must be considered, including the protonation states of reactive residues (histidine, glutamate, aspartic acid, lysine) and disulfide bonds between cysteine residues. In general, these modifications depend on the local chemical environment of the amino acid and are (at least historically) poorly predicted by available computational packages. Visual inspection of SSB reveals that none of it's residues requires special attention.

**Protonation states.** Ordinarily, the reactive residues of a protein must be examined carefully to select appropriate placement of its hydrogen atoms. This is because NAMD does not perform any chemistry and cannot create or destroy covalent bonds.

To build the DNA system, you will first rotate the coordinates of the protein and SSB to coincide with the steered-molecular dynamics (SMD) pulling axis. Change directories to `1-pull-ssb`. We have provided the PDB file `ssb65.pdb`, which was created by connecting DNA bound to SSB in a structure determined through x-ray crystallography (PDB accession code: 1EYG). In VMD, source the file `load-extended-dna.tcl`, which loads `ssb65.pdb`, rotates the coordinates so that the DNA ends lie along the SMD axis, sets the size of the simulation cell, writes the PDB `ssb-oriented.pdb`, and finally extends the ssDNA fully along the SMD axis for subsequent visualization. The **Periodic** tab of the **Graphical Representations** window enables you to show or hide periodic images of DNA. Observe how the DNA will fit between the periodic images of the protein. Note that we assume the DNA will lie along a line as it is removed from the protein. This is approximately true at the rapid rate the DNA is being pulled (150 Å/ns), but there may be unwanted interactions between periodic images. As mentioned above, a better approach to this problem is to truncate the DNA as the simulation progresses so that a small system can be used. However, such an approach is difficult to do with NAMD and VMD and is too advanced for this tutorial to cover.

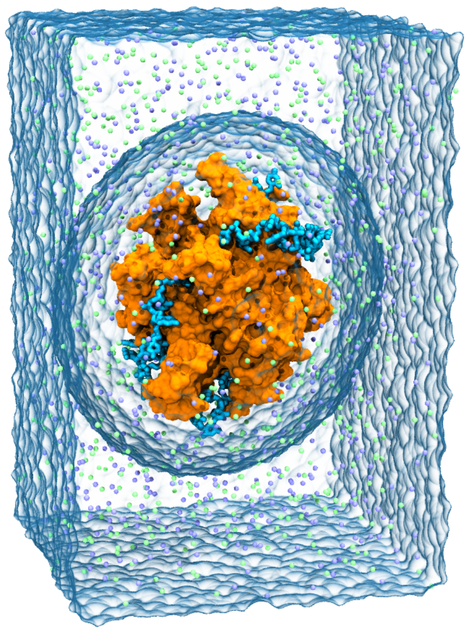


Figure 3: Final system containing $SSB_{35}$. The surface of water added with the solvate plugin of VMD is shown transparently, and indicates the size of the system as well as the size of the solvation shell from Grubmüller's Solvate. Ions are shown as light green and blue vdW spheres. The protein is shown using an orange surface, and the DNA is shown with in cyan with atomic detail.

The system must be constructed in steps, beginning with the coordinates for the protein and DNA in `ssb-oriented.pdb`. The first step is to build a PSF and PDB for the protein and DNA, which will include atoms that were missing from the original PDB (e.g. hydrogen atoms) and add information from the force-field such as the charge. Ordinarily this can be done fairly easily using the graphical AutoPSF plugin of VMD. However, this guide uses the latest version of the CHARMM force-field (CHARMM36), which uses a new patch to convert the default RNA into DNA (more on this later) that breaks the current version of AutoPSF. Thus, this tutorial will guide you through writing a simple Tcl script that is sourced from within VMD to produce the structures. In general, such scripts are more flexible than the built-in graphical interfaces, and also leave a precise record of how your system was built. The build process resembles that depicted in Fig. 4.

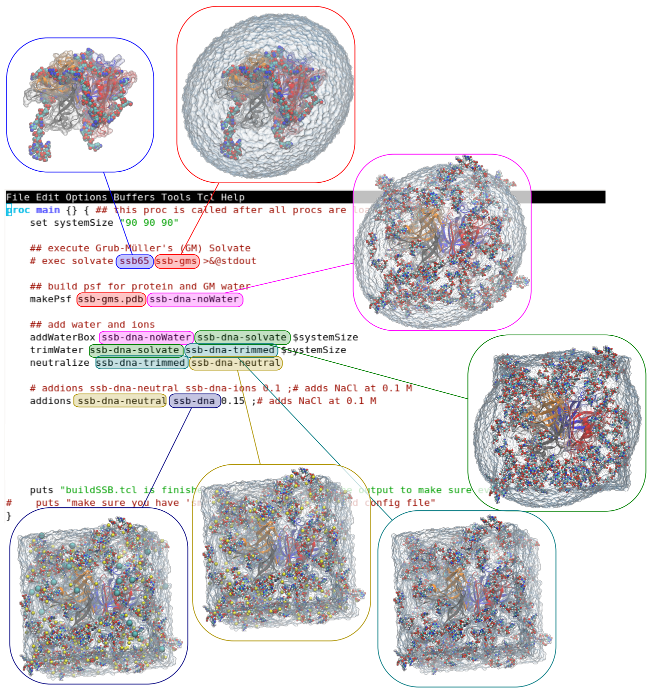
There are three online resources that are particularly useful when writing Tcl scripts for VMD:

1. The Tcl Reference Manual (http://tmml.sourceforge.net/doc/tcl/), which contains information about Tcl commands,
2. the Tcl Text Interface section of the VMD user's guide (http://www.ks.uiuc.edu/Research/vmd/current/ug/node116.html), which explains the extra Tcl commands understood by VMD, and
3. the psfgen User's Guide (http://www.ks.uiuc.edu/Research/vmd/plugins/psfgen/ug.pdf), which describes how the psfgen plugin of VMD can be used.

## Psfgen

The first step towards writing a PSF using a script is to tell VMD to use the psfgen plugin with the command `package require psfgen`. Now, all the psfgen commands are available to the script. The next step is to read in the force-field topology files. These contain information about the atoms in each protein residue or nucleotide, including how they are bonded, what charge they have, and what "types" of atoms they are (this last bit of information is used by NAMD in conjunction with the force-field's parameter file to determine what forces to apply). The topology files can be read with the psfgen command `topology` ***`path/to/file.rtf`***. Your topology files are located in the directory `charmm36.nbfix`. Make your script load all the files with the `.rtf` extension. Finally, make your script load the PDB into VMD with the command `mol new ssb-oriented.pdb`

Now VMD and psfgen are ready to build your protein. This is done piece-by-piece using contiguous "segments" of bonded atoms. For example, SSB is a homotetramer comprised of four (identical) monomers, and each monomer will



```
proc main {} { ## this proc is called after all procs are loa
    set systemSize "90 90 90"

    ## execute Grub-Müller's (GM) Solvate
    # exec solvate ssb65 ssb-gms >&@stdout

    ## build psf for protein and GM water
    makePsf ssb-gms.pdb ssb-dna-noWater

    ## add water and ions
    addWaterBox ssb-dna-noWater ssb-dna-solvate $systemSize
    trimWater ssb-dna-solvate ssb-dna-trimmed $systemSize
    neutralize ssb-dna-trimmed ssb-dna-neutral

    # addions ssb-dna-neutral ssb-dna-ions 0.1 ;# adds NaCl at 0.1 M
    addions ssb-dna-neutral ssb-dna 0.15 ;# adds NaCl at 0.1 M




    puts "buildSSB.tcl is finish                    output to make sure ev
#    puts "make sure you have '                       config file"
}
```

Figure 4: An overview of the system assembly process. A typical compartmentalized system assembly script is depicted. The script was written to evoke explicitly named Tcl procedures that serve as logical wrappers. From top left, clockwise: the system is shown in its initial state, containing protein and DNA with no hydrogen atoms; structured water is added to the DNA and protein using Grubmüller's Solvate program (distinct from VMD's Solvate plugin); the DNA bound to the protein is cut into small pieces that are randomly distributed through the system; VMD's solvate plugin is used to add water that isn't too close to the protein; excess solvent is removed so that the system is a cube; the protein and DNA charge is neutralized as counterions and coions are added to the system at $\sim 1$ M concentratoin using the autoionize plugin of VMD (which also has a graphical interface).

have its own segment. In preparation for adding the segment to psfgen (which has its own memory of the molecule that you are building that is *completely* separate from the molecules loaded into VMD), your script should write a PDB for the first segment.

This can be done in a few lines by first creating an "atom selection" with `set sel [atomselect top "protein and chain A"]`. Here, the atom selection text, "protein and chain A" works just like it would in the **Graphical Representations** window and selects one of the monomers of our protein. The `atomselect` command returns a unique label that can be used as a Tcl command to query or manipulate the selected atoms. When a Tcl command is contained in square brackets, the command is executed and the bracketed-command is substituted with whatever it returns, in this case the unique atomselect label (something like atomselect0 or atomselect1). The code above saved that label in the variable `sel` (this could be called anything) for later use. Usually a PDB from the Protein Data Bank does not have its up-to-four-letter segment defined, but this is crucial for psfgen. The command `$sel set segid APRO` sets the segment name to "APRO" for the selected atoms. Finally `$sel writepdb tmp.pdb` writes the PDB file from the molecule loaded in VMD. Add the three lines described above to your script.

Psfgen has its own memory of the molecules that you are building, and this memory is completely separate from VMD's. Adding or deleting molecules from VMD does not affect psfgen. Similarly, psfgen commands do not alter VMD's state. Thus each monomer must be added to psfgen's picture of the structure using the command:

```
segment unique-segname {
      code-specify-residues-in-segment
}
```

The code argument to `segment` usually contains one psfgen command, `pdb tmp.pdb`, that tells psfgen to extract the bonded information for the segment from a PDB. Note that segment command not load coordinates into psfgen, which must be done after the `segment` command using `coordpdb tmp.pdb`.

At this point, psfgen should have your the first segment molecule in its memory! You can repeat the above steps explicitly writing `tmp.pdb` for the each segment using VMD before adding it to psfgen for the four monomers using suitable atom-selection text. Alternatively, you can generalize your script a little and use a `foreach` loop to repeat the same code for the different chains. This might look like the following:

```
set sel [atomselect top protein]
set chains [lsort -unique [$sel get chain]] ;# return A B C D
foreach chain $chains {
      puts "Adding chain $chain to psfgen"
```

```
        set seg ${chain}PRO
        set sel [atomselect top "protein and chain $chain"]
        $sel set segid $seg
        $sel writepdb tmp.pdb segment $seg { pdb tmp.pdb }
        coordpdb tmp.pdb
}
```

The only new commands are Tcl commands, like `lsort`, which sorts a list (lists are whitespace delineated in Tcl) optionally returning only unique entries, and `foreach`, which executes the loop code once for each element of the list `$chains` after setting the variable `chain` to the value of that element.

Now, write a similar block of code to include the DNA in your structure. At this point, if you wrote the PSF and PDB from psfgen's memory, you would end up with RNA, and not DNA. Between the `segment` and `coordpdb` commands for the DNA, you should "patch" the RNA to turn it into DNA. This is very easy, but requires a loop over each DNA nucleotide. First, create a list of resids with `set resids [lsort -unique [$sel get resid]]`. Then loop over those resids with something like `foreach r $resids { ... }` Inside that loop, apply the patch to the nucleotide "DEOX"[1] with `patch DEOX $seg:$r` . In addition to specifying the bonds, the PSF specifies which atoms should have angles and dihedral angles applied. Unfortunately, the patch statements do not usually specify these correctly, so after applying patches, you should always provide the psfgen command `regenerate angles dihedrals` to automatically reinsert the angles and dihedrals in the PSF.

At this point, the PSF is almost complete. However, the `tmp.pdb` files did not include all the atoms in the structures (namely hydrogen atoms were missing, as is usually the case). Before you write the structure, you should tell psfgen to use the "internal coordinates" specified in the topology file to place the missing atoms in reasonable positions by issuing the command: `guesscoords`. Finally, you are ready to write the PSF and PDB from psfgen's memory using the commands `writepsf` *`psfgen.psf`* and `writepdb` *`psfgen.pdb`* (of course name these files whatever you would like).

Go ahead and source your script from within VMD with `source psfgen.tcl` from the Tk Console. Make sure you look at the resulting structures for anything unusual; this is the trickiest part of building a structure!

## 3.1 Solvent

The next step is to add water to the system. In general, it is a good idea to first use Grubmüllers solvate to add structured water around a protein, but we will forego this step since it requires installation additional external software.

[1]this patch is defined in the topology file for nucleic acids and changes the RNA structure currently in psfgen's memory into DNA

**Careful placement of water.** The structure of water can be influenced 10 Å from a surface, and in this way can act as an extension of the protein. Moreover, the structure of the water around a protein can stabilize its conformation. We typically use a pair of programs called Dowser and Grubmüller's Solvate (accessed from the command line with `dowserx` and `solvate`) to place individual water molecules in energetically favorable locations near the protein in cavities and on the surface, respectively. Note that Grubmüller's Solvate is distinct from the solvate plugin of VMD, which places pre-equilibrated water without considering the interaction between the water and the protein. Both programs are executed prior to creation of the protein structure file. To make this tutorial more portable, these steps have been omitted. However, we recommend employing those pieces of software when building protein systems for production simulations.

Unstructured solvent can be added using the VMD's graphical Solvate plugin, but you can also add two lines to your `psfgen.tcl` script as follows:

```
package require solvate
solvate psfgen.psf psfgen.pdb -minmax "{-57.5 -57.5 -75} {57.5 57.5
75}" -o solvate
```

The numbers in minmax specify the extent to which the solvent should reach, and have been chosen to allow the DNA to move between periodic images of SSB. The usual choice should provide at least 20 Å between periodic images (the structure of water can be affected up to 10 Å from the surface of a protein!) and at least two Debye-lengths (the characteristic length at which ions screen electrostatic interactions, $\sim$ 10 Å in 100 mM monovalent electrolyte such as NaCl, $\sim$ 3 Å in 1 M).

Sometimes `solvate` adds a little too much water, and needs to be trimmed. This doesn't appear to be the case for the 110 Å box used in this tutorial, but you can consult section 2.2 of the psfgen User's Guide for detailed instructions should this problem ever arise.

DNA is highly charged (one negative electron charge per phosphate). Counterions are expected to, more-or-less, neutralize the DNA within a couple of Debye-lengths, so the system should be neutralized before additional ions are added to the appropriate concentration. This is very easily achieved using the graphical Autoionize plugin of VMD, which also has a convenient scripting interface. You can make your script add 1 M with the following commands (the `-sc` option specifies the desired molarity):

```
package require autoionize
autoionize -psf solvate.psf -pdb solvate.pdb -sc 1 -o ssb-dna
```

If you added the `solvate` and `autoionize` commands to your script, open VMD and source your script with `source psfgen.tcl` to execute all the commands. Load the resulting structure and make sure everything looks okay. Check that the system is neutral with the following command: `measure sumweights [atomselect top all] weight charge`. The `measure` command provides a lot of really useful functionality to VMD, especially for analysis of simulation trajectories.

The final step is to flag atoms to apply force using SMD and movingConstraints. A script is provided that does this called `constrainDNA.tcl`. Look at the file and make sure you know what it does. Then source the file from the Tk Console.

## 4 Simulation

Simulations can be performed in the NPT (constant number of atoms, pressure, temperature), NVT (constant number of atoms, volume, temperature), or NVE (constant number of atoms, volume, energy) thermodynamic ensembles. Water is a nearly incompressible fluid, so small changes in the volume cause large changes in the pressure. When building a system, it is almost impossible to obtain a pressure close to atmospheric without simulating in the NPT ensemble. On the other hand, external forces (which in general do not conserve momentum) interact badly with the barostat.

A good approach is to first run a short NPT simulation without external forces until the volume of the system stops changing, then use the volume obtained in the NPT simulation to start an NVT simulation using the correct system size. In this case, the terminal nucleotides must lie along the steered-molecular dynamics (SMD) pulling axis at the onset of the SMD simulation. Because the DNA ends may drift away from their initial positions in the NPT simulation, it is best to start the NVT simulation using the original coordinates rather than the NPT simulation's restart coordinates.

The solvent at the edges of the system may have clashes or small gaps that send shock-waves that perturb the solute conformation. Thus it is best to perform initial equilibration in the NVT simulation with the solute conformation restrained (constrained in NAMD terminology). Once, the system is equilibrated, SMD simulation can begin.

The NPT simulation has already been performed on your behalf. When starting the NVT simulation, you could just use the "extended system" restart file (`.xsc`), but the volume (and pressure) in the NPT simulation fluctuate. A better approach is to find the average the volume during the NPT simulation, and scale your cellBasisVectors accordingly. From the Tk Console, run the following command to extract the average system volume: `source` `averageVolume.tcl`.

Enter the correct cellBasisVectors into `ssb-nvt.namd` and run this locally. This simulation will equilibrate your system with “constraints” (really restraints, but NAMD syntax is not always precisely descriptive) and SMD forces defined (but a value of 0 for the SMD velocity so the ends are merely held in place). Note that the cellBasisVectors are a little smaller than the initial size of the system, and water around the edges will be roughly twice the nominal density when you begin the NVT simulation. Using the speed of sound in water (1500 m/s) to estimate the timescale required for the uneven water density to propagate through the system, you should simulate at least 6.6 ps per 100 Å. While this simulation runs, have a careful look at `ssb-nvt.namd` and `ssb-smd.namd` to make sure you understand the configuration files well. Don’t hesitate to ask questions about the various options.

Once the simulation is finished, you should run `ssb-smd.namd` for a moment to ensure that everything works. `ssb-smd.namd` is the same as `ssb-nvt.namd`, except it uses an SMD velocity of 150 Å/ns and uses the system volume from the `restart.xsc` file if the **`ssb-nvt`** simulation.

## 5 Analysis

Load and examine the SMD simulation trajectory in VMD (`mol new complete/ssb.psf`[2]; `mol addfile complete/output/ssb-smd.dcd` in the Tk Console). Watch how the DNA dissociates from SSB.

In the limit of slow pulling, you can safely assume that the force due to `movingConstraints` is the same as the force due to SMD. In the provided simulation trajectory, the pulling velocity was extremely fast, and the above assumption may not hold Although this may not hold because the pulling velocity was extremely fast in the provided simulations, The only reliable way to extract the force due to `movingConstraints` is to use VMD to track the position of the C1′ atom. However, for simplicity we will assume the movingConstraints and SMD forces have equal magnitudes.

The SMD force can be extracted from a NAMD log file using any number of scripting languages or utilities. If you are comfortable with a particular scripting language (e.g. awk, Perl, or Python), feel free to extract the force from the log file using that language. Presently, you will be guided through this task using Tcl.

The line you are trying to copy from the log file looks like this: `SMD 0 -2.88316 26.5742 -33.5497 150.378 261.271 -3359.08`. This line has the format: “SMD timestep posX posY posZ forceX forceY forceZ”. Create a new

[2]The provided trajectory was built with an old version of the solvate plugin and has a different number of atoms

tcl script called, `getForce.tcl`. First of all, set a variable to the axis direction as follows `set axis [vecnorm "92 115 60"]`. `vecnorm` is a handy VMD command that normalizes a vector. In this file, use the Tcl command `set ch [open complete/output/ssb-smd.log]` to open the file for reading. The `open` command returns a unique channel ID, which you set to the `ch` variable. The command `gets $ch line` will read a line from the file, setting it to the variable `line` and returning the number of characters on the line, or −1 if it reached the end of the file.

To step through the file, you can use a while loop, which executes a conditional statement and then executes the loop code as long as the conditional statement was true ( 0 in Tcl). For example, `while` { `[gets $ch line]` >= 0 } { `puts $line` } would simply copy the contents of the file to the Tk Console. Inside the loop, you must write code that checks if the line begins with "SMD " (note the extra space prevents the line beginning "SMDTITLE" from being printed).

The easiest way to do this is with a very simple *regular expression*.[3] Use an `if` statement and the command `regexp "^SMD " $line` to only print lines that begin with "SMD ". Check to see if it works in the Tk Console by sourcing your script.

There are several ways to extract the relevant information, but probably the easiest employs the `lassign` command to set each whitespace delineated word in the line to a variable. Use this technique to get the vector form of the force. By taking the dot product[4] between the force vector and the axis of pulling, you can obtain the magnitude of the force. Test to see if this works. The units of force printed by SMD are given as kcal/mol Å, but the multiplicative factor 69.48 can be used to convert to piconewtons. To do math in Tcl, you must enclose the mathematical expression a special command like this: `set result [expr 1*4]`. Use this information to convert your force to piconewtons.

Now that you have a basic script, you can open a file for *writing* (rather than reading as we have just done) using the command `set outCh [open` ***`outfile.dat`*** `w]`. It doesn't matter whether the file was pre-existing, but opening a file like will erase its contents. Subsequent commands like `puts $outCh "some text or data"` will print "some text or data" into `outfile.dat`.

Thus, you can print the magnitude of the force along the SMD axis inside the loop to a data file of your choosing. Finally, after the close of the `while` loop, you should close both of the open file channels with the `close $ch` command. Now have a look at the resulting forces using your favorite plotting software!

[3]Regular expressions are implemented in many scripting languages and provide a powerful method for querying and manipulating text. See http://www.tcl.tk/man/tcl8.4/TclCmd/re_syntax.htm for more information about regular expressions in Tcl.

[4]`vecdot $v1 $v2` where `$vN` is a list of numbers like `"1.0 0.0 0.0"`

Note that you will need to employ heavy smoothing to see the signal emerge from the noise. The force that you obtain should be quite large. This is because the pulling velocity was extremely rapid. At a slower rate of 1 Å/ns, the force is on the order of 100 pN, which is still much larger than the forces obtained in experiment. If you have time, modify your script to print the work performed by the SMD spring.

As a side note, if you were to perform this simulation many times, you would be able to apply Jarzynski's equality [3] to obtain an estimate of the free energy change in removing the DNA from SSB from the work performed during the non-equilibrium trajectories.

$$e^{-\beta\,\Delta F} = \overline{e^{-\beta\,W}}$$

The bar denotes an ensemble average; $\beta$ denotes $1/\mathrm{k_B T}$; $\Delta F$ is the change in free energy when the system is brought from one state to another; $W$ is the work done during the change of state. Jarzynski's equality is a relatively recent development in statistical mechanics that has be experimentally validated. We find this development significant because it relates work performed during a non-equilibrium process (performed many times) to an equilibrium property of the system. There are other ways of obtaining free energies from MD simulations, including umbrella sampling, adaptive biasing force, and metadynamics. but we highlight Jarzynski's equality because it has applications in both experiment and simulation.

## References

[1] B. Alberts, A. Johnson, J. Lewis, M. Raff, K. Roberts, and P. Walter. *Molecular Biology of The Cell.* Garland Science, New York & London, 4th edition, 2002.

[2] E. V. Bocharov, A. G. Sobol, K. V. Pavlov, D. M. Korzhnev, V. A. Jaravine, A. T. Gudkov, and A. S. Arseniev. From structure and dynamics of protein L7/L12 to molecular switching in ribosome. *J. Biol. Chem.*, 279:17697–17706, 2004.

[3] C. Jarzynski. Nonequilibrium equality for free energy differences. *Phys. Rev. Lett.*, 78:2690–2693, 1997.

[4] S. Raghunathan, A. Kozlov, T. Lohman, and G. Waksman. Structure of the DNA binding domain of *E. coli* SSB bound to ssDNA. *Nat. Struct. Mol. Biol.*, 7(8):648–652, 2000.

[5] W. Rosche, A. Jaworski, S. Kang, S. Kramer, J. Larson, D. Geidroc, R. Wells, and R. Sinden. Single-stranded DNA-binding protein enhances the stability of CTG triplet repeats in Escherichia coli. *J. Bacteriol.*, 178(16):5042–5044, 1996.