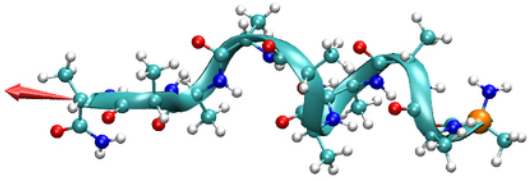
University of Illinois at Urbana-Champaign
Beckman Institute for Advanced Science and Technology
Theoretical and Computational Biophysics Group
Computational Biophysics Workshop

Stretching Deca-alanine



Sanghyun Park

Fatemeh Khalili

April 2009

A current version of this tutorial is available at
http://www.ks.uiuc.edu/Training/Tutorials/

Contents

1 Introduction

In this session, you will be introduced to interactive molecular dynamics (IMD) and steered molecular dynamics (SMD) simulations, and to the calculation of potential of mean force (PMF) from trajectories obtained with SMD simulations.

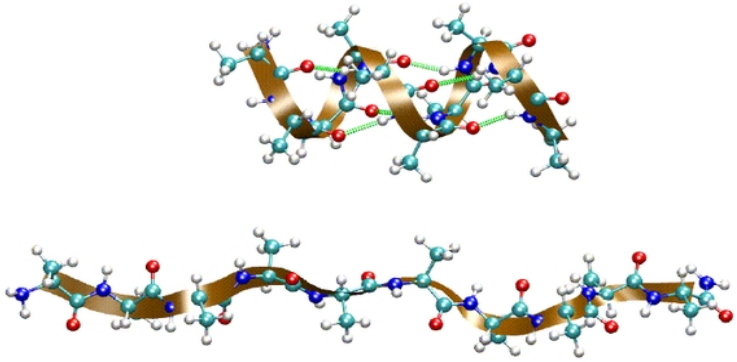


Figure 1: Deca-Alanine in vacuum.

You will be using one system throughout: deca-alanine. Deca-alanine is a peptide composed of ten alanine residues. You will simulate it in vacuum. In vacuum deca-alanine forms an alpha-helix. That is, the alpha-helix is the stable conformation of the molecule in vacuum, as opposed to the beta-strand or the random coil. The helix is shown in the top figure. It is stabilized by six hydrogen bonds (shown in green).

Using IMD and SMD, you will stretch the molecule by applying an external force. As the molecule is stretched, it will undergo a gradual conformational change from the alpha-helix to the random coil (bottom figure). Using SMD trajectories and employing Jarzynski's equality, we will calculate the PMF involved in the helix-coil transition.

2 Setup

A copy of the files needed for these exercises exists in a directory called Workshop in your home directory. In a terminal window, move to this directory by typing:

```
cd ~/Workshop/10Ala-tutorial/files/
```

The content of this directory is shown in Fig. 2 below.

The following programs are used throughout this tutorial.

- VMD – available from http://www.ks.uiuc.edu/Research/vmd/
- NAMD – available from http://www.ks.uiuc.edu/Research/namd/

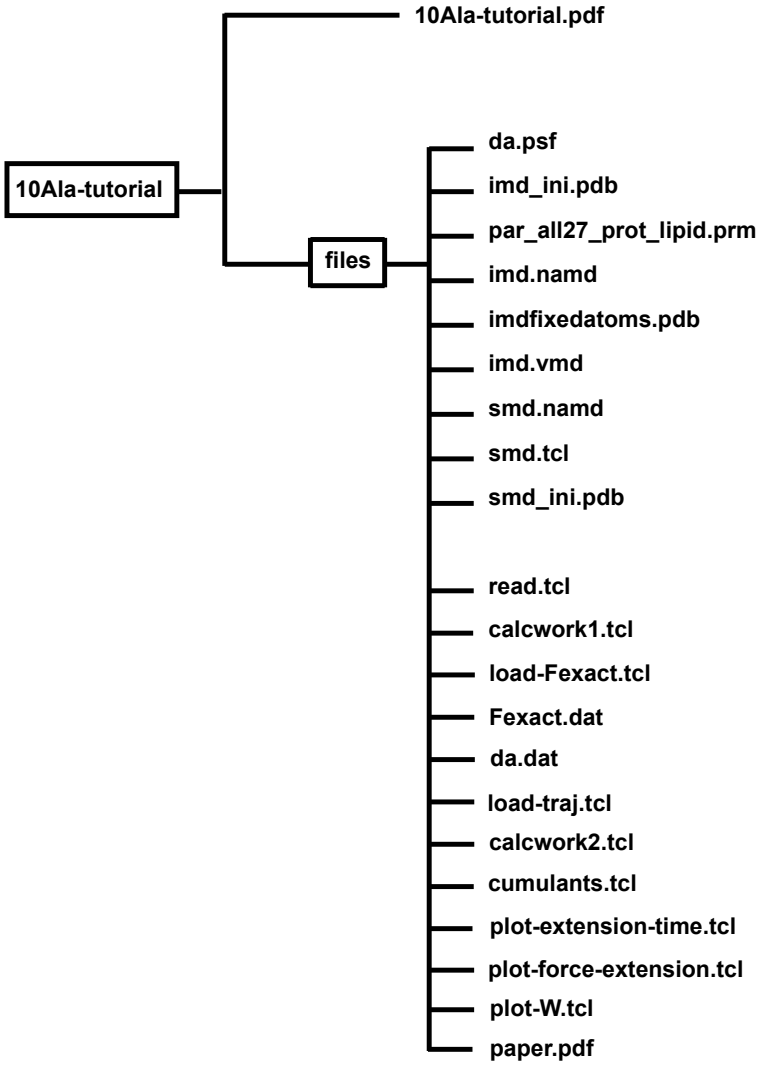


Figure 2: Directory structure for the tutorial exercises.

3 IMD simulation

Let's run an IMD simulation of deca-alanine. Yes, you will be interacting with the simulation.

3.1 1. Starting an IMD simulation

Files needed:

- `da.psf` – protein structure
- `imd_ini.pdb` – initial coordinates
- `par_all27_prot_lipid.prm` – CHARMM parameters
- `imd.namd` – NAMD configuration
- `imdfixedatoms.pdb` – fixed atoms

The following part of imd.namd enables IMD (already in the file):

```
IMDon      on     ;#
IMDport    3000   ;# port number (enter it in VMD)
IMDfreq    1      ;# send every 1 frame
IMDwait    yes    ;# wait for VMD to connect before running?
```

Run NAMD, by typing in a terminal windown:

```
namd2 imd.namd > da_imd.log &
```

The simulation will not actually run until the connection between NAMD and VMD is established.

3.2 2. VMD control of an IMD simulation

Start VMD, in a terminal window type:

```
vmd -e imd.vmd
```

This will load the molecule and show it in both the Ribbon and the CPK representations, as saved in the VMD state file `imd.vmd`. You will see one atom (alpha-carbon of the first residue) colored in orange. That atom will be fixed during the IMD simulation.

Within VMD, select the menu item Extensions → Simulation → IMD Connect (NAMD) .

In the IMD Connection window, enter Hostname: `localhost` and Port: `3000`

Click Connect.

You should see the molecule jiggling as the simulation is running.

Even though the double representation of Ribbon and CPK is good for viewing the overall and detailed structure of the molecule, you may change the representation to whatever you want. You can change the representation while the IMD simulation is running.

3.3 3. Applying a force in an IMD simulation

Now let's apply a force to stretch the molecule.

Select the menu item Mouse → Force → Atom.

Click and drag on an atom located near the other side of the 'orange' atom. The red arrows indicates the force being applied (magnitude and direction). Stretch the molecule, say, by half of its original length.

Now remove the force by middle-clicking on the atom to which the force is being applied.

Observe how the molecule folds back.

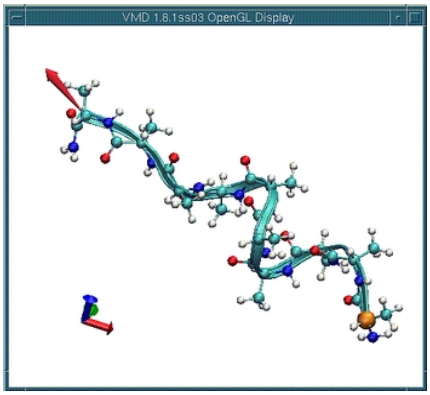


Figure 3: IMD Simulation.

If you want to stop the simulation, click Stop Sim in the IMD Connection window. Otherwise, the simulation is set to run for 100 ps. Try various things: pull

different atoms, squash the molecule, do whatever you want. Use your imagination.

When the simulation stops, the molecule will stop jiggling.

Quit VMD (the menu item File → Quit).

If you want to run the IMD simulation again, repeat the procedure of this section starting with running NAMD.
In another tutorial, you will be introduced to a VMD extension called AutoIMD, which is another way of doing IMD simulations. It automatically generates all the necessary files and launch and connect to a NAMD simulation.

4 SMD simulation

Let's run an SMD simulation of deca-alanine. It is basically a more systematic way of doing the IMD simulation we just finished. IMD simulations are done to explore the system; SMD simulations are done to analyze the system systematically.

Files needed:

- `da.psf` – protein structure
- `smd.namd` – NAMD configuration
- `smd.tcl` – Tcl script
- `par_all27_prot_lipid.prm` – CHARMM parameters
- `smd_ini.pdb` – initial coordinates

The simulation is done at a constant temperature of 300 K. The Langevin dynamics scheme is used for the temperature control. The Tcl script `smd.tcl` is used to apply external forces. Basically the script does the following. One end of the molecule (the N atom of the first residue) is constrained to the origin. The other end (the capping N atom at the C-terminus) is constrained to a point that moves along the z-axis from 13 Å to 33Å with a constant speed of 1 $\AA$/ps. So it takes 20 ps for the full extension. For the constraints, a harmonic potential with a force constant of 7.2 kcal/mol/$\AA^2$ is used.

Now let's run the simulation:

```
namd2 smd.namd > da_smd.log
```

The simulation will run on your local machine. The output will be written to the following files:

`da_smd.log` – standard output
`da_smd.dcd` – trajectory
`da_smd_tcl.out` – Tcl script output

5 Viewing SMD trajectories within VMD

Start VMD loading the trajectory just generated:

```
vmd da.psf da_smd.dcd
```

Position the molecule (rotate, translate, scale) so that you have the view of the entire length of the molecule.

Change the representation: Select the menu item Graphics→ Representations. In the Graphical Representations window, set Drawing Method to Ribbons. The Ribbon representation is good for viewing the overall structure, but you may explore any other representations or even multiple views.

Use the scroll bar at the bottom of the VMD Main window to browse through the trajectory.

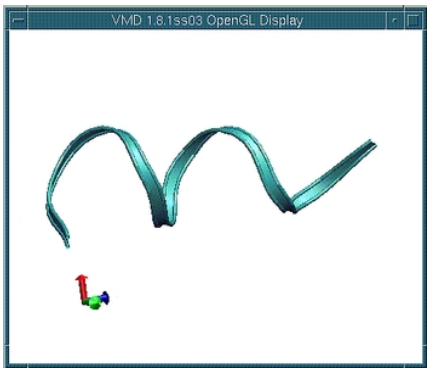


Figure 4: Deca-Alanine in ribbon representation, loaded into VMD.

An important structural change during the helix-coil transition is the breaking of hydrogen bonds. You can monitor hydrogen bonds using VMD.

Choose CPK from Drawing Methods. This will change the representation of the molecule from Ribbon to CPK.

Now let's show hydrogen bonds:

1. In the Graphics Representations window, click Create Rep.

2. In the Selected Atoms text entry, delete the word `all`, type `name N O`, and hit the Enter key. (Hydrogen bonds are formed between N and O atoms.)

3. Select HBonds from Drawing Method.

4. Change the parameters to the following: Distance Cutoff: `4.0`, Angle Cutoff: `40`, Line Thickness: `10`.

You should see several hydrogen bonds.

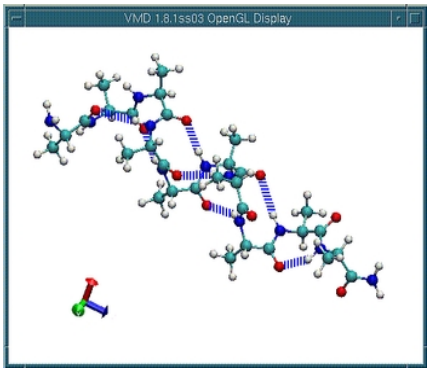


Figure 5: Deca-alanine in CPK representataion. Hydrogen-Bonds are highlighted.

Again using the scroll bar at the bottom of the VMD Main window, browse through the trajectory. Observe the hydrogen bond breaking as the molecule is stretched.

Once you are done, quit VMD (the menu item File → Quit).

6 Analysis of the SMD trajectory

Launch VMD by typing `vmd` in a terminal window.

Within VMD select the menu item Extension → Tk Console. A console window should appear with a prompt. The following Tcl/Tk commands are executed from within the TkCon window. We will use the tag `>>` to indicate Tcl commands.

Make sure you are in the correct directory by typing:

```
>> cd ~/Workshop/10Ala-tutorial/files/
```

Open the Tcl Output file generated by your SMD simulation, and store its content to a variable `mytraj`:

```
>> set infile [open da_smd_tcl.out r]
>> set mytraj [read $infile]
>> close $infile
```

The content of the file `da_smd_tcl.out` is printed on the screen. You should see three columns:

First column: time (ps)
Second column: extension, or the end to end distance (Å)
Third column: applied force (kcal/mol/Å)

We use the following units: ps for time, Å for distance, kcal/mol for energy.

Assign columns of the `mytraj` to three separate arrays (lists), by running the following Tcl script:

```
>> source read.tcl
```

The script defines three variables, `t, z, f`, corresponding to each column of `mytraj` respectively.

Define parameters, v (pulling velocity) and dt (time step of the data):

```
>> set v 1
>> set dt 0.1
```

Recall that one end of the molecule was constrained to the origin and the other end was constrained to a moving point. The distance between the two constraints was changed from 13 Å to 33 Å with the constant velocity v. Define an array (list) c representing the distance between the two constraint points at each time step:

```
>> set c {}
>> set i 13
>> while {$i < 33.1} { lappend c $i; set i [expr $i + $v * $dt]  }
```

Plot $z - t$ and $c - t$ curves using the `multiplot` plugin within VMD. In the TkCon window type:

```
>> package require multiplot
>> set plot1 [multiplot -x $t -y $z -plot]
>> $plot1 add $t $c -plot
```

Q: The extension z generally lags behind c, the distance between the two constraints. How big is the lag in this case? What would you do if you want to reduce the lag further?

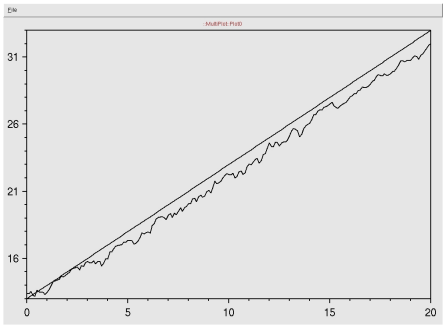


Figure 6: Extension versus time during the simulation, compared with the distance between the constraint point (straight line).

Now plot the force-extension curve:

```
>> set plot2 [multiplot -x $z -y $f -plot]
```

Q: Is the force generally positive or negative? Why?

The work done on the system during the pulling simulation is:

$$W(t) = v \int_0^{t'} dt' f(t') \tag{1}$$

where v is the pulling velocity.

To calculate the work w by numerical integration, run the following Tcl script:

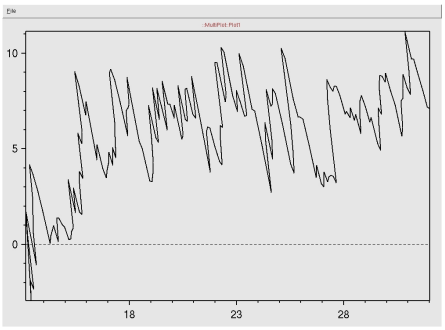


Figure 7: Force versus extension during the simulation.

```
>> source calcwork1.tcl
```

The work is stored in a variable (list) called w.

If a pulling simulation is performed very slowly, then the process is reversible. The work done during such a reversible pulling is equal to the free energy difference of the system between initial and final states. Thus, a reversible work curve can be considered as an exact PMF. For our system, the reversible pulling speed turns out to be about 0.0001 Å/ps (10000 times slower than the one you used).

The exact PMF obtained from a reversible pulling is stored in the file `Fexact.dat`. Read the content of the file, and store it in a variable called `Fexact`, by running the follwong Tcl script:

```
>> source load-Fexact.tcl
```

The array `Fexact` contains two columns, representing the potenial of mean force (second column) at each extension (first column).

Plot $W - c$ in blue together with $Fexact$ in red:

```
>> set plot3 [multiplot -x $c -y $w -linecolor blue -plot]
>> $plot3 add $Fexact(1) $Fexact(2) -linecolor red -plot
```

Q: How do they compare? What is the origin of the difference?

Quit VMD:

```
>> quit
```

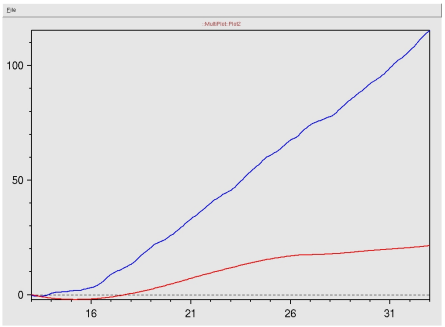


Figure 8: Work versus extension in the fast pulling simulation (blue) and reversible pulling simulation (red).

6.1 PMF calculation

The trajectory you have generated in this session is actually not practical for the PMF calculation because the pulling speed was too high. Besides, you need multiple trajectories to use Jarzynski's equality. Therefore, you will use pre-generated trajectories.

Launch VMD by typing `vmd` in a terminal window.

Within VMD select the menu item Extensions → Tk Console. The following commands are all executed from within the `TkCon` window.

Make sure you are in the correct directory by typing:

```
>> cd ~/Workshop/10Ala-tutorial/files/
```

Ten trajectories obtained from SMD simulations with a pulling speed of 0.01 Å/ps are stored in `da.dat`. This pulling speed is 100 times slower than the one you used, but still 100 times faster than the reversible speed. Load the trajectories as well as the exact PMF:

```
>> source load-Fexact.tcl
>> source load-traj.tcl
```

The script `load-traj.tcl` defines three variable. The array t contains time steps of the data. z and f are matrices of 10 columns, with each column representing the extension and the applied force for each trajectory, respectively.

To plot the extension-time curve for each trajectory, you need to reduce the number of data points plotted for each graph. The following script plots z vs t for a tenth of the data points. Each trajectory is plotted with a different color:

```
>> source plot-extension-time.tcl
```

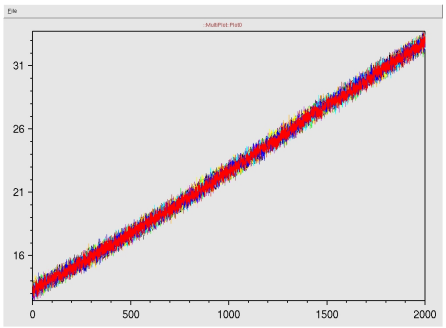


Figure 9: Extension versus time for 10 pulling trajectories.

And plot the force-extension curve:

```
>> source plot-force-extension.tcl
```

In each figure window, you should see ten overlapping curves, with each curve corresponding to a trajectory.

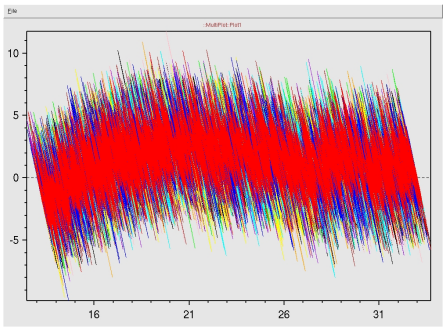


Figure 10: Force versus extension for 10 pulling trajectories.

Define parameters: dt, time step; v, pulling speed; T, temperature (including Boltzmann's constant).

```
>> set dt 0.1
>> set v 0.01
>> set T 0.6
```

Define an array c representing the distance between the two constraint points at each time step:

```
>> set c {}
>> set i 13
>> while {$i < 33.001} {lappend c $i; set i [expr $i + $v * $dt] }
```

Calculate work for each trajectory by executing the following command:

```
>> source calcwork2.tcl
```

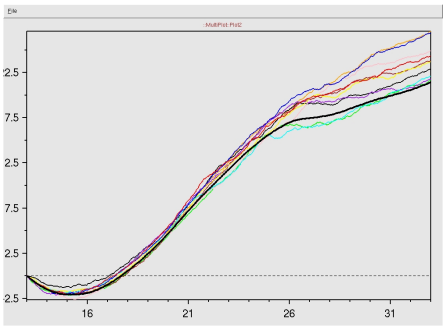


Figure 11: Work versus extension for 10 pulling trajectories. Fexact, obtained from reversible pulling is shown in bold.

Plot W vs. c together with the exact PMF:

```
>> source plot-W.tcl
```

Plot the exact PMF on the same graph:

```
>> $plot3 add $Fexact(1) $Fexact(2) -linewidth 3 -linecolor black -plot
```

You should see ten work curves and one curve for the exact PMF. The thick black line is the exact PMF.

Q: Work done to the system should be on average higher than the PMF. Are all the work curves higher than the PMF, or do you also see some work curves lower than the PMF?

Let's now employ Jarzynski's equality:

$$\exp\{\frac{-[F(c(t)) - F(c(0))]}{T}\} = \ < \exp[-W(t)/T] > \qquad (2)$$

or

$$F(c(t)) - F(c(0)) = -T \log < \exp[-W(t)/T] > \quad (3)$$

The right-hand side can be expanded in terms of so-called cumulants:

$$F(c(t)) - F(c(0)) =< W(t) > -\frac{1}{2T}[< W(t)^2 > -< W(t) >^2] + \cdots \quad (4)$$

where the cumulants up to the second order are shown.

Estimate the PMF according to three different formulas: the original formula (exponential average), the first order cumulant expansion formula (average work), and the second order cumulant expansion formula. The following Tcl script will calculate and store the three different estimates in `Fexp, F1, F2` variables.

```
>> source cumulants.tcl
```

Plot the three estimates for the PMF together with the exact PMF. Plot the exact PMF, Fexact, in red, the average work (or the first order cumulant expansion), $F1$, in green, the second order cumulant expansion, $F2$, in blue, and the exponential average, Fexp, in black:

```
>> set plot4 [multiplot -plot]
>> $plot4 add $Fexact(1) $Fexact(2) -linecolor red -plot
>> $plot4 add $c $F1 -linecolor green -plot
>> $plot4 add $c $F2 -linecolor blue -plot
>> $plot4 add $c $Fexp -linecolor black -plot
```

Q: Which estimate is the closest to the exact PMF? Does it make sense to you in view of what you learned in the lecture?

Quit VMD:

```
>> quit
```

For more information on the PMF calculation from SMD simulations, see the paper included `(paper.pdf)`.

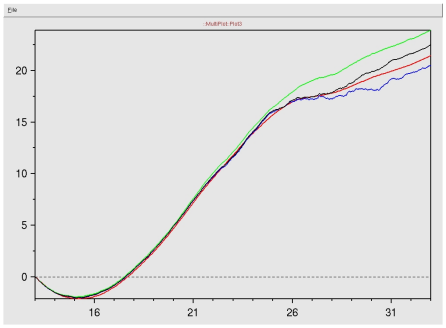


Figure 12: The PMF obtained from the cumulant expansion of Jarzynski's equality, compared with the PMF obtained from reversible pulling simulation (shown in red).