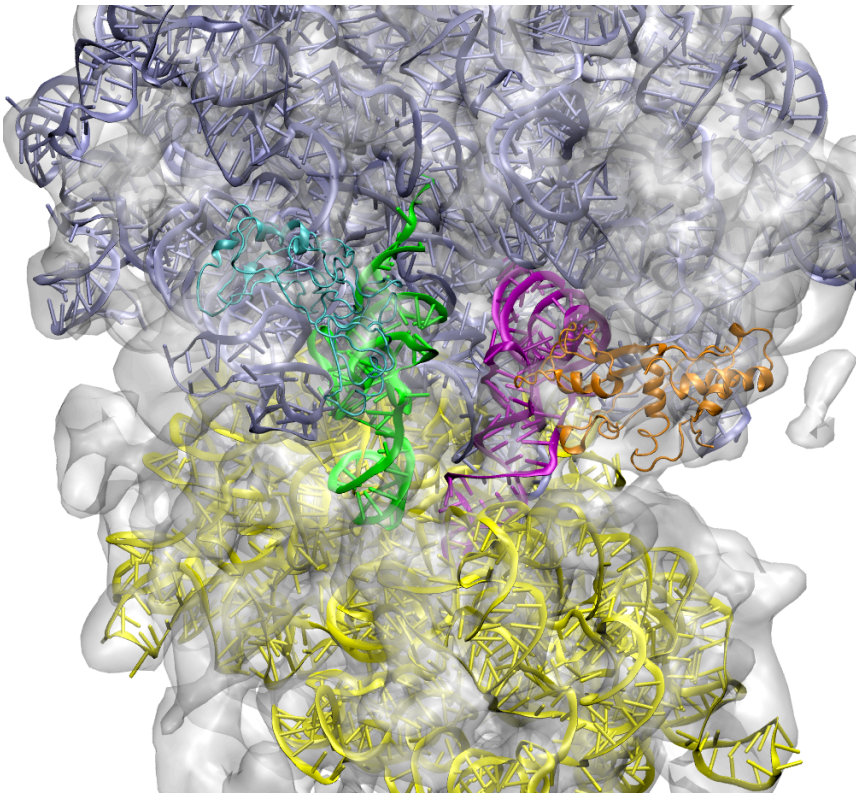
UPPSALA
UNIVERSITET

MMB 3.0



Tutorial

Updated November 4, 2019

Copyright and Permission Notice

Acknowledgments

Samuel Flores's early development of RNABuilder was funded by the Simbios National Center for Biomedical Computing through the National Institutes of Health Roadmap for Medical Research, Grant U54 GM072970. Information on the National Centers can be found at http://nihroadmap.nih.gov/bioinformatics. Past support has been received from eSSENCE and Uppsala University. The author is currently Dean of the Swedish National Graduate School in Medical Bioinformatics, at Stockholm University, funded by a Swedish Research Council grant.

Table of Contents

1 OVERVIEW 9

2 PREREQUISITES AND INSTALLATION INSTRUCTIONS 11

3 EXERCISE 0: YOUR FIRST MMB RUN 15

3.1 Objectives 15
3.2 Verify you have the required files 15
3.3 Open a command prompt/terminal window 16
3.4 Navigate to your MMB folder 16
3.5 Run MMB 17
3.6 Visualize MMB results 18

4 EXERCISE 1: GENERATING YOUR FIRST 3D MODEL 21

4.1 Objectives 21
4.2 Examining and editing the input parameters file 21
4.2.1 RNA and protein sequence commands 22
4.2.2 Stage parameters 22
Run parameters 23
4.2.3 Temperature 25
4.2.4 Base pairing and nucleic acid duplex force commands 25
4.2.5 Global simulation parameters 28
4.2.6 Turning on the MD force field 28

5 EXERCISE 1B: GENERATING YOUR FIRST 3D MODEL, MODERN WAY WITH NTC'S 31

5.1 Objectives 31
5.2 Examining and editing the input parameters file 31
5.2.1 RNA and protein sequence commands 32
5.2.2 Stage parameters 32
Run parameters 33
5.2.3 Temperature 35
5.2.4 Base pairing and nucleic acid duplex force commands 35
5.2.5 Global simulation parameters 38

5.2.6 Turning on the MD force field *38*
5.2.7 Turning OFF the old-style stacking parameters *39*

6 EXERCISE 2: READING STRUCTURES FROM A PDB FILE AND RIGIDIFYING PARTS OF YOUR MODEL 41

6.1 Objectives 42
6.2 Rigid mobilizers and Weld constraints 42
6.3 SelectedAtoms 44
6.4 Run example 45
6.5 On your own: Determine the effects of the Weld constraints and rigidification 45
6.6 On your own: Turn the RNA into a different 3D structure 46

7 EXERCISE 3: HOMOLOGY MODELING OF RNA 47

7.1 Objectives 47
7.2 Specifying the template 47
7.2.1 Read in the PDB file for the template *47*
7.2.2 Specify template sequence to match information in the PDB file *48*
7.2.3 Rigidify the template *48*
7.3 The target chain 48
7.3.1 Specify the sequence of the target chain *48*
7.3.2 Account for sterics using "Physics where you want it" *48*
7.4 Apply forces to pull the corresponding residues together 49
7.5 Run example 50

8 EXERCISE 4: PROTEIN HOMOLOGY MODELING 52

8.1 Objectives 52
8.2 Specifying the template 52
8.2.1 Provide the PDB file for the template *52*
8.2.2 Start the run *53*
8.2.3 Specify template sequence to match information in the PDB file *53*
8.2.4 Specify model sequence, for which no structural information is available *53*
8.2.5 Rigidify the template and constrain it to ground *53*
8.2.6 Globally align the model and template backbones (with gaps) *54*
8.2.7 Locally align based on fragments (no gaps) *55*

9 EXERCISE 5: PROTEIN MORPHING 57

9.1 Objectives 57

9.2 Introduction 57
9.3 Preparing the input structure file 57
9.4 Start the run 58
9.5 Examine the input file 58
9.6 On your own: complete the morph with a fully-flexible alignment 61

10 EXERCISE 6: TEMPLATE-BASED DOCKING 63

10.1 Objectives 63
10.2 Introduction 63
10.3 Input structures 64
10.4 Start the run 64
10.5 Examine the input file 64
10.6 On your own: docking a different antigen 66

11 EXERCISE 6: EFFICIENTLY GENERATE ALTERNATE PROTEIN CONFORMATIONS 67

11.1 Objectives 67
11.2 Introduction 67
11.3 The command file 68
11.4 Run MMB 68
11.5 Analyze the conformational coverage 68

12 ZEMU: ZONE EQUILIBRATION OF MUTANTS 71

12.1 Objectives 71
12.2 Introduction 71
12.3 Preparing the command file 73
12.4 Evaluating binding energy using FoldX 76
12.5 Challenge: find mutations that lower the binding energy even further 77

13 EXERCISE 7: SOLVE PROTEIN STRUCTURE BY NMR CONSTRAINTS 78

13.1 Objectives 78
13.2 Instructions 78
13.3 Results 82

14 EXERCISE 8: FITTING TO ELECTRON DENSITY MAPS 83

14.1 Objectives 83
14.2 Introduction 83
14.3 Run MMB 84

14.4 The command file 85
14.5 View the results 87
14.6 On your own 87

15 VIRTUAL ASSEMBLY OF A PROTEIN-DNA COMPLEX 89

15.1 Objectives 89
15.2 Introduction 89
15.3 Run MMB 89
15.4 The command file 90
15.5 View the results 92
15.5.1 Symmetry expansion with PyMOL 92
15.5.2 Symmetry expansion using other tools 93

16 GRAPHICAL USER INTERFACE WITH CHIMERA 94

16.1 Manual install of the plugin for Chimera 94
16.1.1 Manual install on Mac OSX 94
16.1.2 Manual install on Linux 95
16.1.3 Manual Install on Windows 95
16.2 Using the GUI 95
16.2.1 RNA folding example 96
16.2.2 Flexible Protein morphing 98

1 Overview

MMB constructs 3D structural and dynamical models of RNA and protein by applying user-specified base pairing interactions, interatomic forces, sterics, bond mobilities, and structural constraints. The forces, constraints, mobilities, parameters, and molecules can change from one simulation stage to another. It uses multi-resolution techniques, such as coarse-grained force fields and selective rigidification of groups of atoms, to decrease computation time.

MMB is run from the command line and requires a user-provided input parameter file that specifies the simulation, and optionally an input structural coordinate file in PDB format. It produces trajectory files, also in PDB format.

MMB was written in C++ code using Simbody and its molecular modeling extension, Molmodel. Binary installations are available for Windows, Intel-based Mac, and Linux. The source code is also freely available for download.

For a summary of what is new in this release, and for citation info, please see the Reference Guide.

2 Prerequisites and installation instructions

You will need VMD (or equivalent) plus the MMB executable and auxiliary files to do the exercises in this manual:

1. **VMD (or another software for viewing trajectory files):** We have experienced some problems installing VMD 1.8.7 on Windows. VMD 1.8.6 or Pymol can also be used.

   To install VMD, go to http://www.ks.uiuc.edu/Research/vmd. Click on "Download VMD" and select the installation for your platform. Follow the on-line instructions for installing.

2. **MMB:** MMB is a tool for building 3D RNA and protein models using a variety of knowledge the user has about the structure. It runs from the command line (don't expect a graphical interface!) and requires a user-generated input file and an MMB parameter file (more details below). The main output of MMB is the trajectory it generates from the provided parameters, in PDB format. The last frame of the trajectory is also saved as a PDB file.

   You can download MMB from http://simtk.org/home/rnatoolbox. Click on "Downloads." You should now be at: https://simtk.org/project/xml/downloads.xml?group_id=359 . Follow the directions for your operating system below to install and test MMB.

   Windows:

For Windows, download `Installer.2_14.Windows.zip`. Find it in Windows Explorer (a Windows Explorer window probably opened automatically when you downloaded the package). Right-click on `Installer.2_14.Windows.zip` and click on "`Extract to the specified folder`" in the mouse menu. In the "`Destination path`," type "`C:\Users\Installer.2_14.Windows`", or some other path of your choice. Don't extract it in "Program Files," because on some Windows flavors this directory does not allow writing output files. Version 2.4 is not yet available for Windows, so Windows users will not be able to do all of the examples, in particular Exercise 8 (fitting to density maps).

MacOSX:

If you are running Leopard or later, download and save the latest available installer file, ending with `.tgz`. Move this file to another location – we suggest your home directory, in mac that would be `/Users/[your-user-name]`, in Linux that would be `/home/[your-user-name]`, or you can just use the Linux/Unix shortcut "~"; that's what we will do in this tutorial.

Now, open a Terminal window. You will find the Terminal application in:
Macintosh HD -> Applications -> Utilities -> Terminal
You will now decompress the above file. In Terminal, issue e.g.:

```
cd ~
tar -zxvf Installer.2_18.OSX.tgz
```

The files will be decompressed into the `~/Installer.???.[OSX | Ubuntu]` directory.

Now, you just need to tell OSX where to find the library files. That's easy, all the MMB files are in the same directory. You can specify this in your `~/.bash_profile` or wherever you put your configuration file, or just do it manually every time you run MMB. In bash the command is:

```
export     DYLD_LIBRARY_PATH=/Users/[your-user-
name]/Installer.2_18.OSX/lib

export     LD_LIBRARY_PATH=/home/[your-user-
name]/Installer.2_18.Ubuntu/lib
```

Note that you will have to adjust the above depending on where you installed MMB. Note also that you can't use the "~" shortcut in your `.bash_profile`.

Make sure you issue `source ~/.bash_profile` if you went that route. Now you're ready to go!

Linux (32 Bit):

If you are using Linux, download and save the file:

```
Installer.2_18.Ubuntu.tgz
```

If you have a different distribution of Linux, download this file anyway and let me know if it works; otherwise we may have to install from source.

Decompress by issuing:

```
tar –zxvf Installer.2_18.Ubuntu.tgz
```

This will unpack into a directory called `Installer.2_14.[OSX | Ubuntu]`. Move this directory to the location of your choice. In this tutorial we will assume you moved it into ~.

You should be in the bash shell. You can figure out what shell you're using by issuing `echo $SHELL` . If you're not in bash, issue : `bash` .

Now, you just need to tell Linux where to find the library files. That's easy, all the MMB files are in the same directory. You can specify this in your `~/.bash_profile` or wherever you put your configuration file, or just do it manually every time you run M. In bash the command is:

```
export LD_LIBRARY_PATH=~/Installer.2_18.Ubuntu
```

.. or wherever you installed the package. Make sure you issue `source ~/.bash_profile` if you went that route. Now you're ready to go!

Linux (64 Bit):

The instructions are mostly the same as for 32-bit Linux. Except you will download:

```
Installer.2_14.Linux64.tgz
```

Again unzip it in ~. Then:

```
export LD_LIBRARY_PATH=~/Installer.2_14.Linux64
```

It may also search for liblapack.so.3 and libblas.so.3, which appear to go under the name lib*.so.3gf on some debian derivates. Some of our users have reported that this is dealt with by just making symlinks with the .so.3 suffixes for MMB to run. Throughout this tutorial, just follow the Linux 32-Bit instructions, issuing `MMB.2_14.Linux64` instead of `MMB` .

3 Exercise 0: Your first MMB run

3.1 Objectives

This first exercise is intended for you to:

- Learn how to invoke MMB
- Verify that your installation is working properly
- Learn how to visualize the M-generated trajectory within VMD

3.2 Verify you have the required files

MMB requires two files in order to run:

- *parameters.csv*: This is a parameter file, analogous to those used by molecular dynamics programs to set bond, stretch, bend, torsion, etc. parameters. One of the main differences is that the parameters.csv specifies the rotation and translation relating the glycosidic nitrogen atoms in interacting pairs of bases. Casual users are unlikely to modify this file.

  On Windows, this file should have been copied from the latest examples folder into your MMB folder.

- An input file: This text file specifies the sequence, base pairs, and any other constraints, forces, and options that should be applied. In this exercise, we will use *commands.singlebasepair.dat*.

Verify that you have these two files in your MMB folder. The default/recommended locations for your MMB folder are:

(Windows) My Computer -> C: -> Users -> Installer.2_14.Windows
(Mac OSX) ~/Installer.2_18.OSX
(Linux) ~/Installer.2_18.Ubuntu

3.3 Open a command prompt/terminal window

MMB is run from the command prompt/terminal/console. If you haven't already done so, to launch a command prompt/terminal window, select:

(Windows) Start -> All Programs -> Accessories -> Command Prompt
(Mac OS) Macintosh HD -> Applications -> Utilities -> Terminal
(Linux) Open the Console that comes with your distribution.

3.4 Navigate to your MMB folder

For these exercises, we will be running MMB from the MMB folder. It is possible to run MMB from other directories, but the directory must contain the *parameters.csv* parameter file.

Within the command prompt/terminal window/console, navigate to the MMB folder. If you installed in the default locations, you would type:

(Windows) `cd "C:\Users\Installer.2_14.Windows"`
(Mac OSX) `cd ~/Installer.2_18.OSX`
(Linux) `cd ~/Installer.2_18.Ubuntu`

Note: Quotation marks are required in specifying directory paths within the Windows command prompt window if the directory path includes spaces.

3.5 Run MMB

To run MMB, type:

(Windows) `MMB.2_14.exe -c commands.singlebasepair.dat`
(Mac OS) `./MMB -c commands.singlebasepair.dat`
(Linux) `./MMB -c commands.singlebasepair.dat`

For OSX, you have a choice of executables depending on what version you're using. The `-c` option specifies the input file name, in this case *commands.singlebasepair.dat*.

Note: If you do not specify the `-c` option, MMB uses a default input file name of *commands.dat*.

You should see output that looks something like this:

```
[TwoTransformForces.cpp] Satisfied contacts   : 0 out of : 1
Writing structure for reporting interval # 1
[TwoTransformForces.cpp] Satisfied contacts   : 0 out of : 1
Writing structure for reporting interval # 2
[TwoTransformForces.cpp] Satisfied contacts   : 0 out of : 1
Writing structure for reporting interval # 3
...
```

If instead you see messages like the following:

```
/Users/Sam/svn/RNAToolbox/trunk/src/ParameterReader.cpp:2312 Unable
to open command file: commands.singlebasepair.dat
```

MMB could not find the input file you specified (in this case *commands.singlebasepair.dat*). Make sure you spelled the file name correctly and that it exists in the directory from which you are calling MMB.

3.6 Visualize MMB results

MMB generates a number of files that by default are saved to the directory from which you ran MMB. You should see a *last.1.pdb* file and a *trajectory.1.pdb* file. *last.1.pdb* is the PDB file for the last frame in the trajectory, which is typically the most interesting for structure prediction. The entire trajectory is saved in NMR format in the file *trajectory.1.pdb*.

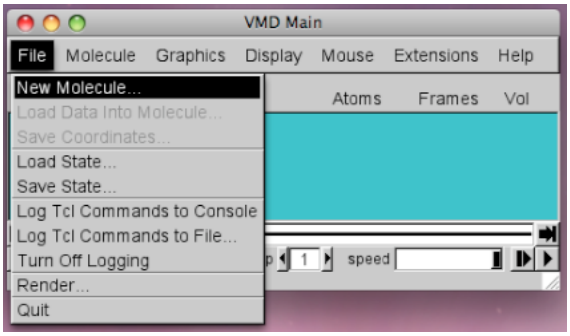
We can visualize the resulting trajectory within VMD:

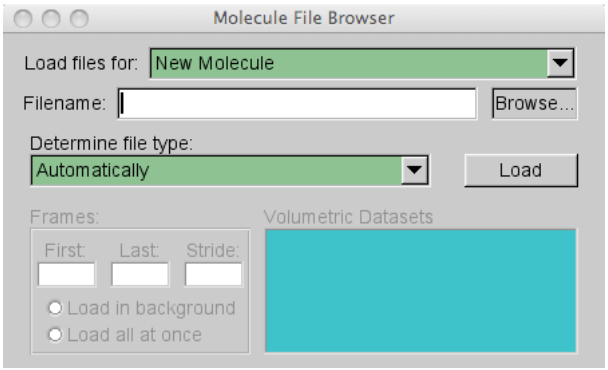
1. Launch VMD. If you installed VMD in typical locations, you would select:

   (Windows) Start -> All Programs -> University of Illinois -> VMD -> VMD 1.8.7
   (Mac OS) Macintosh HD -> Applications -> VMD
   (Linux) The location may depend on your distribution.

2. The "VMD Main" window will appear. Select:

   File -> New Molecule...





3. In the "Molecule File Browser" that appears, click on "Browse" and select the *trajectory.1.pdb* created by MMB. The location for this file (for default/recommended installations) is:

   (Windows) My Computer -> C: -> Users -> Installer.2_14.Windows
   (Mac OSX) ~/Installer.2_18.OSX
   (Linux) ~/Installer.2_18.Ubuntu

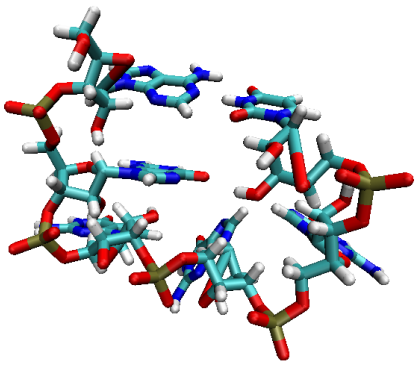
4. Change the molecule representation by going to the "VMD Main" window and selecting:

   Graphics -> Representations

   From the drop-down menu for "Drawing Method," select "Licorice." (In VMD 1.8.7, you might want to try the "New Cartoon" method, which provides a nice visualization of the molecule).

5. You should see a structure like that shown below by the end of the trajectory (the "Licorice" drawing method was used). In this simple example, a single base pair was specified pulling the two ends together.

   To rotate the structure, click and drag the structure. To translate the structure, type `t` and then click and drag the structure. To return to rotating the structure, type `r`.



4 Exercise 1: Generating your first 3D model

4.1 Objectives

In Exercise 1, you will:

- Learn about some of the key parameters that need to be specified within the command file
- Use your new knowledge about MMB to build a GNRA tetraloop from a starting sequence and published geometric constraints

4.2 Examining and editing the input parameters file

To edit or create and input parameters files, you must use a text editor (NOT a program like Microsoft Word, which will add many hidden characters for formatting, etc.) For Windows, we recommend using WordPad (Start -> All Programs -> Accessories -> WordPad). On Mac, some options include vi, emacs, and TextEdit (Macintosh HD -> Applications -> TextEdit.app).

Start your text editor and open up the file *commands.hairpin-short.dat*, located in your MMB folder. The default locations for your MMB folder are:

(Windows) My Computer -> C: -> Users -> Installer.2_14.Windows
(Mac OSX) ~/Installer.2_18.OSX
(Linux) ~/Installer.2_18.Ubuntu

The syntax of the input parameters file is that each row contains information about one particular parameter. The first word in the row is the name of the parameter, followed by one or more values needed to specify that parameter.

4.2.1 RNA and protein sequence commands

The first section of the *commands.hairpin-short.dat* file contains the sequence parameters, described below. The `baseInteraction` records (discussed later) must appear sometime *after* the `firstResidueNumber` of each interacting chain in the base pair has been specified. `firstResidueNumber` must appear sometime *after* the corresponding `sequence` has been specified. Other than that, the order in which parameters are listed usually does not matter, except in some advanced usages not covered in this tutorial.

```
RNA A 2656 UACGUAAGUA
```

To instantiate a biopolymer, use `RNA`, `DNA` or `Protein` command. This takes the following parameters: chain ID (string, single character long), first residue number (integer), and sequence (string, single letter code).
This example instantiates an RNA chain with chain identifier "A", first residue number 2656, and the sequence shown in single-letter code. The chain identifier should be a single character in compliance with the PDB format. The sequence can be quite long, dependent mostly on your available memory. If you are supplying an input PDB structure file, the coordinates will be matched according to the chain ID and residue number.

4.2.2 Stage parameters

MMB can divide up the simulation into stages, each with its own set of simulation parameters. This allows flexibility in how the simulation is performed. Stages are explained in more detail in Exercise 2. In this exercise, we will not be dividing up the simulation into stages, so the first stage and the last stage are the same:

```
firstStage 1
lastStage 1
```

This starts the simulation at stage 1, and ends when stage 1 is over.

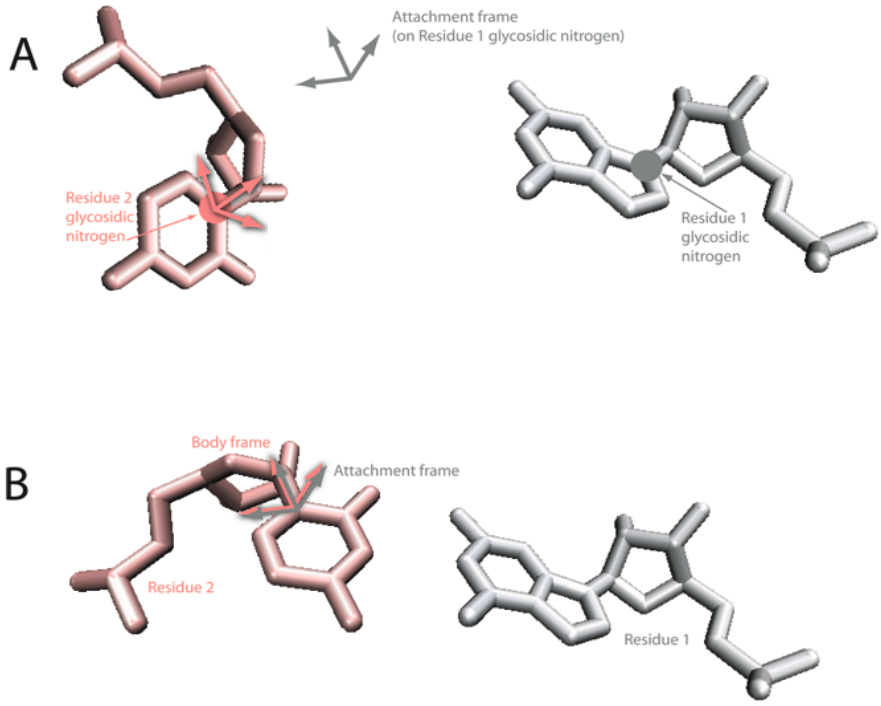
Run parameters

The next section of the *commands.hairpin-short.dat* file specifies the run parameters, which control the bookkeeping aspects of the MMB simulation.

4.2.2.1 baseInteractionScaleFactor

The `baseInteractionScaleFactor` (alias `forceMultiplier`, `twoTranformForceMultiplier`) is a scaling factor applied to the `baseInteraction` forces and energies. The base pairing forces themselves are applied using the following scheme.

First, an *attachment frame* is generated which is part of the first residue's glycosidic nitrogen body, but located outside it. Then a *body frame* is generated which is located at the center of the second residue's glycosidic nitrogen. The *body frame*'s x-axis points along the glycosidic bond, and its z-axis is perpendicular to its base plane. The location and orientation of the *attachment frame* is such that when it is aligned with the second residue's *body frame* the desired base pairing geometry is attained. Thus the task of parameterizing the MMB force field is firstly that of determining the correct position and orientation of the *attachment frame*. We distribute a program to compute this given the coordinates of a base pair with the desired geometry, but will not cover its use in this tutorial. After this is done one must also determine the depth of the potential well and its range – again beyond the scope of this tutorial.



In this exercise, `baseInteractionScaleFactor` was set to `20`, to make the forces strong enough for convergence:

```
baseInteractionScaleFactor 200
```

Note that it is not good idea to make the force multiplier *too* strong, because this will make the system stiff, which means there will be fast oscillations which will in turn require the variable time step integrator to take small time steps. If the `baseInteractionScaleFactor` parameter is not specified, it defaults to 1.

4.2.2.1 reportingInterval

This parameter controls the frequency of trajectory frames (reporting intervals) written by MMB.

`reportingInterval 4.0`	This instructs MMB to output a trajectory frame for every 4.0 ps of simulation time, starting at time 0

4.2.2.2 numReportingIntervals

This parameter controls the number of such frames written by MMB at a single stage.

Clearly, simulation time = numReportingIntervals * reportingInterval .

`numReportingIntervals 10`	This instructs MMB to write 10 frames at the applicable stage.

4.2.3 Temperature

In the *commands.hairpin-short.dat* file, the `temperature` parameter is specified:

`temperature 10.0`	This sets the temperature of the simulation to 10.0

If set`Temperature` is set to `TRUE`, as it is by default, MMB uses one of several available thermostat algorithms (set by `thermostatType`, which defaults to `NoseHoover`) to hold the system temperature to this setpoint. Note that thermostats do not conserve system energy.

4.2.4 Base pairing and nucleic acid duplex force commands

To generate base pairing forces to form the stem you can use the command:

```
nucleicAcidDuplex     <chain identifier A>
                      <first residue on A>
                      <last residue on A>
                      <chain identifier B>
                      <first residue on B>
                      <last residue on B>
```

Recalling that the duplex is antiparallel, we require that:

`(first residue on A) < (last residue on A)`

and

`(first residue on B) > (last residue on B)`

In the *commands.hairpin-short.dat* file, you will see an example of this:

```
nucleicAcidDuplex A 2656 2658 A 2665 2663
```

You can also specify the base pairing forces explicitly and individually. The syntax is:

```
baseInteraction <chain identifier for first residue>
           <residue number for first residue>
           <interacting edge for first residue>
           <chain identifier for second residue>
           <residue number for second residue>
           <interacting edge for second residue>
           <glycosidic bond orientation>
```

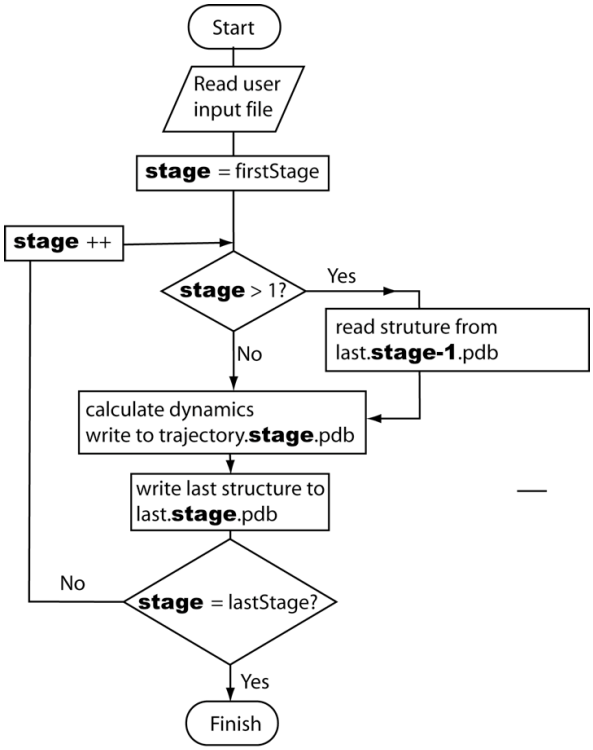
You can create the three base pairing forces above in this alternative way:,

```
baseInteraction A 2658 WatsonCrick  A 2663 WatsonCrick Cis
baseInteraction A 2657 WatsonCrick  A 2664 WatsonCrick Cis
baseInteraction A 2656 WatsonCrick  A 2665 WatsonCrick Cis
```

The first line specifies an interaction between the Watson-Crick edges of residues `2658` and `2663` of chain A, with the glycosidic bonds in the `Cis` orientation. See *Appendix: Forces* for an explanation of this type of interaction. See the same appendix for the other supported combinations of base pairing parameters.

When MMB sees three or more `WatsonCrick/WatsonCrick/Cis` interactions applied to three consecutive residues on each of two strands, it will automatically apply stacking interactions (`HelicalStackingA3/HelicalStackingA5/Cis`) to the consecutive residues (in this case `2656-2657, 2657-2658, 2663-2664,` and `2664-2665`). Thus the total number of `baseInteraction`'s in the system is 3+4 = 7. MMB monitors how many of these are approximately satisfied at each reporting interval, as you will see.

4.2.4.1 Using stages



MMB divides the simulation into stages, each with its own set of simulation parameters. The first stage is run using information solely from the input parameters file. Since there is no structure, all biopolymers are instantiated as extended chains. The last structure in this stage is written out to the file *last.1.pdb*. This *last.1.pdb* is the starting structure for stage 2 of the simulation. Similarly, at the end of stage 2, the file *last.2.pdb* is written out and used as the starting structure for stage 3. This process repeats for as many stages as specified.

Note that we can use this to start MMB using *any* PDB structure file. In the above explanation, `firstStage` was set to `1`, but there's nothing stopping us from setting it to a higher stage and reading in an arbitrary structure file, as follows:

1. Renaming the desired PDB file to *last.1.pdb*. Make sure the chain ID and residue numbering in the PDB file match that in the command file.
2. Setting the parameter `firstStage` to `2`
3. `lastStage` would also need to be greater than or equal to `2`

But let's not do that now! It will be part of a future exercise.
For now, we will use stages to change our simulation parameters, as we explain next.

4.2.4.2 Turning any parameter into a staged parameter

Any parameters or commands can be enclosed in `readAtStage ... readBlockEnd` tags. This means that the enclosed parameters will be read only for the specified stage. So if you want to read certain values for parameter1, parameter2, etc only during stages 3, do this:

```
readAtStage 3
parameter1 value1
```

```
parameter2 value2
…
readBlockEnd
```

You can have as many of these blocks as you wish, and use them to change the parameters at many stages. There are some other block markers which behave differently (e.g. `readFromStage, readToStage, readUntilStage, readExceptAtStage`), see the *Reference guide*. Also, there are a few nuances to keep track of. The input file is read from top to bottom. Parameters encountered more than once in the input file are overwritten with the one closer to the bottom of the file prevailing. Commands (e.g. `baseInteraction`), on the other hand, are additive, rather overwriting each other. See also Appendix: Forces.

In this tutorial the first stage is very short – we are creating a hairpin without concern for steric clashes:

```
reportingInterval 4.0
numReportingIntervals 10
```

So we are specifying that at stage 1, we will run for 10 reporting intervals. Note that total simulation time = `numReportingIntervals*reportingInterval` .. so for stage 1 simulation time is 40 ps.

4.2.5 Global simulation parameters

The next section of the *commands.hairpin-short.dat* file specifies global simulation parameters, properties that apply to the overall simulation.

`numReportingIntervals 10`	This determines how many frames are generated. In this case, 10 intervals are requested, resulting in 11 frames (if we count last.1.pdb as the 11th) representing a 40-ps simulation.

4.2.6 Turning on the MD force field

You will see the following macro in your input file:

`setDefaultMDParameters`

This turns on all the PARM99 force field terms (except GBSA). It's equivalent to setting the following parameters:

```
globalAmberImproperTorsionScaleFactor     1
globalBondBendScaleFactor                 1
```

```
globalBondStretchScaleFactor              1
globalBondTorsionScaleFactor              1
globalCoulombScaleFactor                  1
globalVdwScaleFactor                      1
globalGbsaScaleFactor                     0
.
```

4.2.6.1 Run example

In your command prompt/terminal window (see Exercise 0), type:

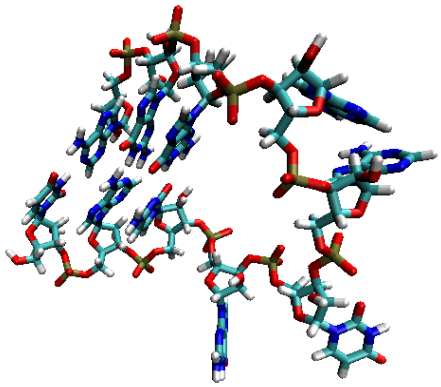
(Windows) `dir`
(Mac OS, Linux) `ls`

You will see a list of files in your current directory. Make sure you have *commands.hairpin-short.dat* and *parameters.csv*. If not, navigate to the directory with these files (see Exercise 0).

Now, run this example by typing:

(Windows) `MMB.2_14.exe -c commands.hairpin-short.dat`
(Mac OS) `./MMB -c commands.hairpin-short.dat`
(Linux) `./MMB -c commands.hairpin-short.dat`

The trajectory from this simulation run is in *trajectory.1.pdb*. The "1" in the output file name refers to the fact that the results are from stage 1. This trajectory can be loaded into and visualized with VMD (see Exercise 0. Make sure you first restart VMD or delete the molecule you loaded in that exercise). By the end of the trajectory, you should see a structure like that shown below. Notice how the 3 base pairs at the ends of the chain have been enforced to produce the hairpin structure.



5 Exercise 1B: Generating your first 3D model, modern way with NtC's

5.1 Objectives

This version of exercise 1 uses Nucleotide Conformers, which yield much better results, particularly with regard to helices. In this exercise, you will:

- Learn about some of the key parameters that need to be specified within the command file
- Learn about Nucleotide Conformers (NtC's)
- Use your new knowledge about MMB to build a GNRA tetraloop from a starting sequence and published geometric constraints

5.2 Examining and editing the input parameters file

To edit or create and input parameters files, you must use a text editor (NOT a program like Microsoft Word, which will add many hidden characters for formatting, etc.) For Windows, we recommend using WordPad (Start -> All Programs -> Accessories -> WordPad). On Mac, some options include vi, emacs, and TextEdit (Macintosh HD -> Applications -> TextEdit.app).

Start your text editor and open up the file *commands.hairpin-short.dat*, located in your MMB folder. The default locations for your MMB folder are:

(Windows) My Computer -> C: -> Users -> Installer.2_14.Windows
(Mac OSX) ~/Installer.2_18.OSX
(Linux) ~/Installer.2_18.Ubuntu

The syntax of the input parameters file is that each row contains information about one particular parameter. The first word in the row is the name of the parameter, followed by one or more values needed to specify that parameter.

5.2.1 RNA and protein sequence commands

The first section of the *commands.GNRA-NtC.dat* file contains the sequence parameters, described below. The `baseInteraction` records (discussed later) must appear sometime *after* the `firstResidueNumber` of each interacting chain in the base pair has been specified. `firstResidueNumber` must appear sometime *after* the corresponding `sequence` has been specified. Other than that, the order in which parameters are listed usually does not matter, except in some advanced usages not covered in this tutorial.

`RNA A 2639 UACGUAAGUA`	To instantiate a biopolymer, use `RNA`, `DNA` or `Protein` command. This takes the following parameters: chain ID (string, single character long), first residue number (integer), and sequence (string, single letter code). This example instantiates an RNA chain with chain identifier "A", first residue number 2639, and the sequence shown in single-letter code. An experimental structure of this is PDB ID 5MRC, with the same residue numbering. The chain identifier should be a single character in compliance with the PDB format. The sequence can be quite long, dependent mostly on your available memory. If you are supplying an input PDB structure file, the coordinates will be matched according to the chain ID and residue number.

5.2.2 Stage parameters

MMB can divide up the simulation into stages, each with its own set of simulation parameters. This allows flexibility in how the simulation is performed. Stages are explained in more detail in Exercise 2. In this exercise, we will not be dividing up the simulation into stages, so the first stage and the last stage are the same:

```
firstStage 1
lastStage 1
```

This starts the simulation at stage 1, and ends when stage 1 is over.

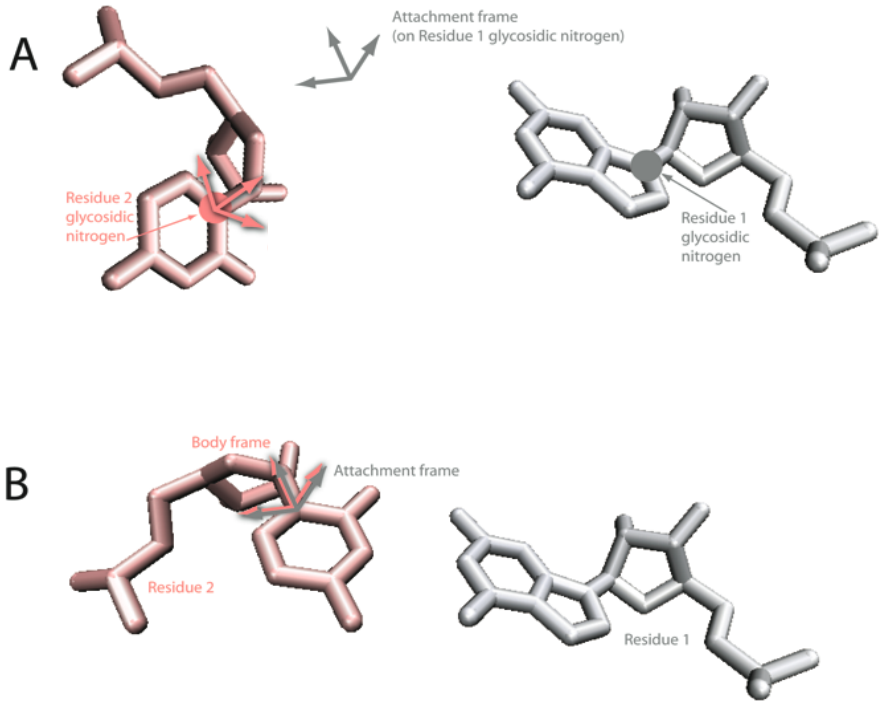
Run parameters

The next section of the *commands.GNRA-NtC.dat* file specifies the run parameters, which control the bookkeeping aspects of the MMB simulation.

5.2.2.1 forceMultiplier

The `forceMultiplier (`alias `twoTranformForceMultiplier, baseInteractionScaleFactor`) is a scaling factor applied to the `baseInteraction` forces and energies. The base pairing forces themselves are applied using the following scheme.

First, an *attachment frame* is generated which is part of the first residue's glycosidic nitrogen body, but located outside it. Then a *body frame* is generated which is located at the center of the second residue's glycosidic nitrogen. The *body frame*'s x-axis points along the glycosidic bond, and its z-axis is perpendicular to its base plane. The location and orientation of the *attachment frame* is such that when it is aligned with the second residue's *body frame* the desired base pairing geometry is attained. Thus the task of parameterizing the MMB force field is firstly that of determining the correct position and orientation of the *attachment frame*. We distribute a program to compute this given the coordinates of a base pair with the desired geometry, but will not cover its use in this tutorial. After this is done one must also determine the depth of the potential well and its range – again beyond the scope of this tutorial.



In this exercise, `baseInteractionScaleFactor` was set to `20`, to make the forces strong enough for convergence:

```
baseInteractionScaleFactor 200
```

Note that it is not good idea to make the force multiplier *too* strong, because this will make the system stiff, which means there will be fast oscillations which will in turn require the variable time step integrator to take small time steps. If the `baseInteractionScaleFactor` parameter is not specified, it defaults to 1.

5.2.2.2 reportingInterval

This parameter controls the frequency of trajectory frames (reporting intervals) written by MMB.

`reportingInterval 3.0`	This instructs MMB to output a trajectory frame for every 4.0 ps of simulation time, starting at time 0

5.2.2.3 numReportingIntervals

This parameter controls the number of such frames written by MMB at a single stage.

Clearly, simulation time = numReportingIntervals * reportingInterval .

`numReportingIntervals 6`	This instructs MMB to write 10 frames at the applicable stage.

5.2.3 Temperature

In the *commands.hairpin-short.dat* file, the `temperature` parameter is specified:

`temperature 10.0`	This sets the temperature of the simulation to 10.0

If set`Temperature` is set to `TRUE`, as it is by default, MMB uses one of several available thermostat algorithms (set by `thermostatType`, which defaults to `NoseHoover`) to hold the system temperature to this setpoint. Note that thermostats do not conserve system energy.

5.2.4 Base pairing and nucleic acid duplex force commands

To generate base pairing forces to form the stem you can use the command:

```
nucleicAcidDuplex     <chain identifier A>
                      <first residue on A>
                      <last residue on A>
                      <chain identifier B>
                      <first residue on B>
                      <last residue on B>
```

Recalling that the duplex is antiparallel, we require that:

`(first residue on A) < (last residue on A)`

and

`(first residue on B) > (last residue on B)`

In the *commands.GNRA-NtC.dat* file, you will see an example of this:

```
nucleicAcidDuplex A 2656 2658 A 2665 2663
```

You can also specify the base pairing forces explicitly and individually. The syntax is:

```
baseInteraction <chain identifier for first residue>
           <residue number for first residue>
           <interacting edge for first residue>
           <chain identifier for second residue>
           <residue number for second residue>
           <interacting edge for second residue>
           <glycosidic bond orientation>
```

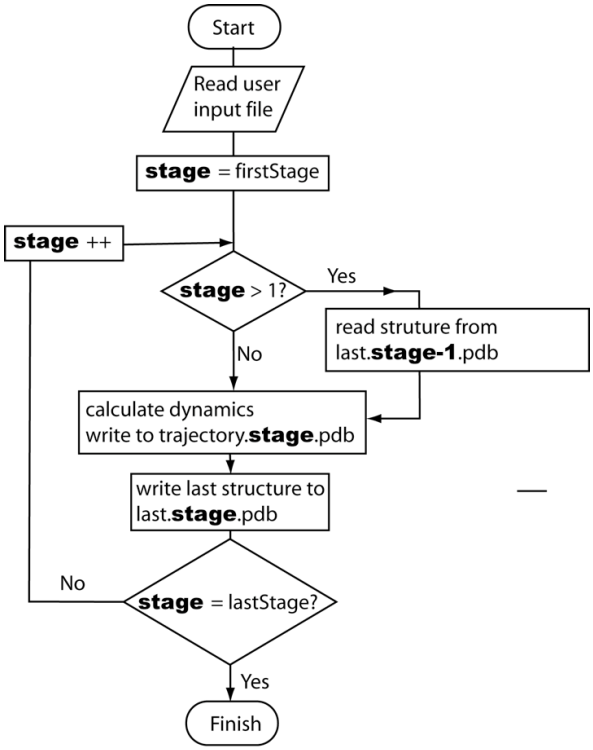
You can create the three base pairing forces above in this alternative way:,

```
baseInteraction A 2658 WatsonCrick  A 2663 WatsonCrick Cis
baseInteraction A 2657 WatsonCrick  A 2664 WatsonCrick Cis
baseInteraction A 2656 WatsonCrick  A 2665 WatsonCrick Cis
```

The first line specifies an interaction between the Watson-Crick edges of residues `2658` and `2663` of chain A, with the glycosidic bonds in the `Cis` orientation. See *Appendix: Forces* for an explanation of this type of interaction. See the same appendix for the other supported combinations of base pairing parameters.

When MMB sees three or more `WatsonCrick/WatsonCrick/Cis` interactions applied to three consecutive residues on each of two strands, it will automatically apply stacking interactions (`HelicalStackingA3/HelicalStackingA5/Cis`) to the consecutive residues (in this case `2656-2657, 2657-2658, 2663-2664,` and `2664-2665`). Thus the total number of `baseInteraction`'s in the system is 3+4 = 7. MMB monitors how many of these are approximately satisfied at each reporting interval, as you will see.

5.2.4.1 Using stages



MMB divides the simulation into stages, each with its own set of simulation parameters. The first stage is run using information solely from the input parameters file. Since there is no structure, all biopolymers are instantiated as extended chains. The last structure in this stage is written out to the file *last.1.pdb*. This *last.1.pdb* is the starting structure for stage 2 of the simulation. Similarly, at the end of stage 2, the file *last.2.pdb* is written out and used as the starting structure for stage 3. This process repeats for as many stages as specified.

Note that we can use this to start MMB using *any* PDB structure file. In the above explanation, `firstStage` was set to `1`, but there's nothing stopping us from setting it to a higher stage and reading in an arbitrary structure file, as follows:

4. Renaming the desired PDB file to *last.1.pdb*. Make sure the chain ID and residue numbering in the PDB file match that in the command file.
5. Setting the parameter `firstStage` to `2`
6. `lastStage` would also need to be greater than or equal to `2`

But let's not do that now! It will be part of a future exercise.
For now, we will use stages to change our simulation parameters, as we explain next.

5.2.4.2 Turning any parameter into a staged parameter

Any parameters or commands can be enclosed in `readAtStage ... readBlockEnd` tags. This means that the enclosed parameters will be read only for the specified stage. So if you want to read certain values for parameter1, parameter2, etc only during stages 3, do this:

```
readAtStage 3
parameter1 value1
```

```
parameter2 value2
…
readBlockEnd
```

You can have as many of these blocks as you wish, and use them to change the parameters at many stages. There are some other block markers which behave differently (e.g. `readFromStage, readToStage, readUntilStage, readExceptAtStage`), see the *Reference guide*. Also, there are a few nuances to keep track of. The input file is read from top to bottom. Parameters encountered more than once in the input file are overwritten with the one closer to the bottom of the file prevailing. Commands (e.g. `baseInteraction`), on the other hand, are additive, rather overwriting each other. See also Appendix: Forces.

In this tutorial the first stage is very short – we are creating a hairpin without concern for steric clashes:

```
reportingInterval 3.0
numReportingIntervals 6
```

So we are specifying that at stage 1, we will run for 10 reporting intervals. Note that total simulation time = `numReportingIntervals*reportingInterval` .. so for stage 1 simulation time is 18 ps.

5.2.5 Global simulation parameters

The next section of the *commands.hairpin-short.dat* file specifies global simulation parameters, properties that apply to the overall simulation.

`numReportingIntervals 6`	This determines how many frames are generated. In this case, 10 intervals are requested, resulting in 11 frames (if we count last.1.pdb as the 11th) representing a 40-ps simulation.

5.2.6 Turning on the MD force field

You will see the following macro in your input file:

`setDefaultMDParameters`

This turns on all the PARM99 force field terms (except GBSA). It's equivalent to setting the following parameters:

```
globalAmberImproperTorsionScaleFactor      1
globalBondBendScaleFactor                  1
```

```
globalBondStretchScaleFactor              1
globalBondTorsionScaleFactor              1
globalCoulombScaleFactor                  1
globalVdwScaleFactor                      1
globalGbsaScaleFactor                     0
.
```

5.2.7 Turning OFF the old-style stacking parameters

We used to use baseInteraction's to impose the correct stacking geometry in helices. These got applied automatically whenever three or more WatsonCrick base pairs were imposed in a row. Let's turn that off:

```
setHelicalStacking False
```

Now there is a more modern and effective way to do this, by restraining the backbone for consecutive pairs of nucleic acid residues. NtC class AA00 is the most populated class, corresponding to A-form helices. Let's impose those on the three base pairs in the helix, starting with the first stretch:

```
NtC A 2639 2640 AA00 .5
NtC A 2640 2641 AA00 .5
```

And also on the complementary stretch:

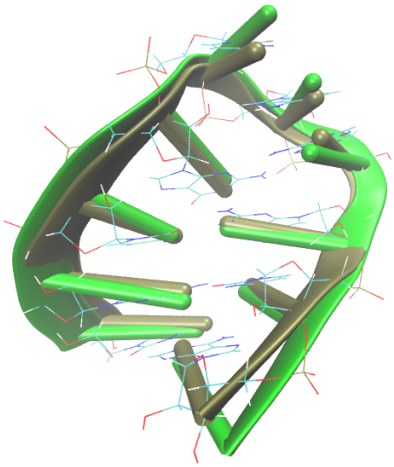
```
NtC A 2646 2647 AA00 .5
NtC A 2647 2648 AA00 .5
```

5.2.8 Make the loop GNRA

In order to make a GNRA tetraloop, we have to specify a few interactions in the loop. First, we must create the so-called sheared base pair between the G and A of the GNRA:

```
readAtStage 2
```

```
# You can upload the provided 5MRC.GNRA.pdb to dnatco.org and see
which Nucleotide Conformers (NtC's) are needed to recapitulate the
GNRA tetraloop. You should find:
NtC A 2641 2642 AA00 1.5
NtC A 2642 2643 OP03 1.5
NtC A 2643 2644 AA08 1.5
NtC A 2644 2645 AA00 1.5
NtC A 2645 2646 AA05 1.5
# Since we are specifying the conformation of every single residue,
we no longer need the non-bonded MD terms:
globalCoulombScaleFactor 0.0
globalVdwScaleFactor 0.0
readBlockEnd
```



5.2.8.1 Run example

In your command prompt/terminal window (see Exercise 0), type:

(Windows)	`dir`
(Mac OS, Linux)	`ls`

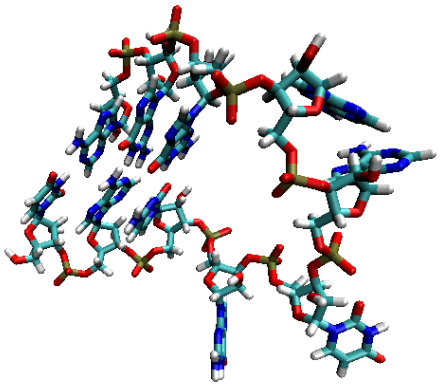
You will see a list of files in your current directory. Make sure you have *commands.hairpin-short.dat* and *parameters.csv*. If not, navigate to the directory with these files (see Exercise 0).

Now, run this example by typing:

(Mac OS) `./MMB -c commands.GNRA-NtC.dat`

(Linux) `./MMB -c commands.GNRA-NtC.dat`

The trajectory from this simulation run is in *trajectory.1.pdb*. The "1" in the output file name refers to the fact that the results are from stage 1. This trajectory can be loaded into and visualized with VMD (see Exercise 0. Make sure you first restart VMD or delete the molecule you loaded in that exercise). By the end of the trajectory, you should see a structure like that shown below. Notice how the 3 base pairs at the ends of the chain have been enforced to produce the hairpin structure.



6 Exercise 2: Reading structures from a PDB file

and rigidifying parts of your model

6.1 Objectives

In this exercise, you will:

- Learn how to use stages to read in and simulate a structure from a PDB file
- Learn about two new types of constraints that can be applied to your model: Weld and Rigid
- Experiment to see what happens when you release these constraints

In your MMB folder, you should see the following files: *1ARJ.short.pdb* and *commands.TAR.dat*. If you do not see them, make sure you are in your MMB folder (see Exercise 0).

1ARJ.short.pdb is the file that we want MMB to read in, so let's copy it to a file named *last.1.pdb*. In your command prompt/terminal window, type:

(Windows) `copy 1ARJ.short.pdb last.1.pdb`
(Mac OS, Linux) `cp 1ARJ.short.pdb last.1.pdb`

Now, let's look at the input parameters file. Open *commands.TAR.dat* in your text editor. Notice that `firstStage` and `lastStage` are both set to `2`. Notice also that `sequence` and `firstResidueNumber` are set to match that of the TAR molecule. (You can compare the values for these parameters with the PDB entry for 1ARJ at http://www.pdb.org/pdb/explore/remediatedSequence.do?structureId=1ARJ).

6.2 Rigid mobilizers and Weld constraints

MMB allows you to (1) fix a chain to ground, (2) weld two residues to each other, and (3) rigidify continuous stretches of residues.

(1) is useful for instances when you are not interested in the overall rotation and translation of a molecule, or when you expect that the 5' end would be fixed in an experimental situation. (2) is often useful when two strands of a helix have been made rigid and now need to be fixed with respect to each other, or to fix the ends of a flexible loop to each other. (3) can be used, for example, to rigidify regions of a molecule to focus resources on a small region of interest, or to model the motion of domains about a flexible hinge.

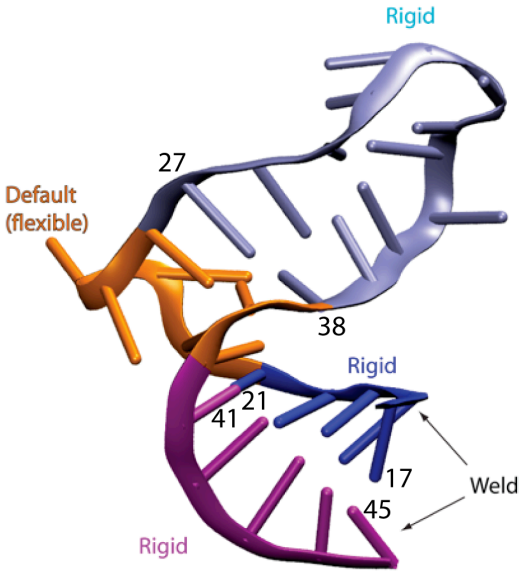
Note that while (3) almost always saves computer time, (1) and (2) may actually increase it. The reason for this is that rigidification involves Rigid *mobilizers,* but welding specifies Weld *constraints*. The latter create constraint equations which must then be satisfied, while the former simply prevent internal degrees of freedom from being created in the first place. See the *Simbody* literature (http://simtk.org/home/simtkcore and look under "Documents") for details on this. Also note that there are many more ways to control the bond mobilities in M, which we will not discuss in this tutorial.

The following parameter settings in the *commands.TAR.dat* show how to set up these different types of rigidification. Refer to the diagram on the next page for the residue numbering.

`removeRigidBodyMomentum False`	By default, MMB removes the rigid body momenta and keeps the system center of mass at the origin. While this is useful to prevent the system from spinning or drifting, it is not compatible with constraints to Ground, so we will turn it off for this simulation.
`constrainToGround N 17`	This command fixes the C3' atom of the specified residue (here chain N, residue 17) to the ground frame. See *Appendix: Forces* for an alternative command.
`mobilizer Rigid N 17 21` `mobilizer Rigid N 41 45`	These two lines rigidify helix I except for the base pair adjacent to the bulge (residues 17 to 21 and residues 41 to 45).
`mobilizer Rigid N 27 38`	The stretch of residues from 27 to 38 (most of helix II plus the loop) are rigidified.

```
constraint N 17 Weld  N 45

constraintTolerance .001
```

This line welds the two ends (residues 17 and 45) together.
This line controls the fidelity with which Rigid and Weld constraints are enforced. A value of .001 means that all internal coordinates must be fixed within .001 nanometers or radians, depending on whether they are distances or angles.



6.3 SelectedAtoms

We use a syntactical variation of the `contact` command introduced in Exercise 1. Here we use the `SelectedAtoms` scheme, and also specify the residue range explicitly:

```
contact SelectedAtoms N FirstResidue LastResidue
```

Where `FirstResidue` and `LastResidue` are self explanatory – but we could just as easily have given residue numbers (including any insertion codes) explicitly – see the *Reference guide*.

These parameters you've encountered before:

```
numReportingIntervals  200
reportingInterval 2.0
firstStage 2
lastStage 2
temperature   10.0
```

6.4 Run example

Make sure you are still in the directory with the *commands.TAR.dat* and the *parameters.csv* files. To do this, in your command prompt/terminal window (see Exercise 0), type:

(Windows) `dir`
(Mac OS, Linux) `ls`

You will see a list of files in your current directory. Make sure you have *commands.TAR.dat* and *parameters.csv*. If not, navigate to the directory with these files (see Exercise 0).

Now, run this example by typing:

(Windows) `MMB.2_14.exe -c commands.TAR.dat`
(Mac OS) `./MMB -c commands.TAR.dat`
(Linux) `./MMB -c commands.TAR.dat`

The trajectory from this simulation run is in *trajectory.2.pdb* file. Note the "2" in the file name. Since the first stage in this run was "2," the corresponding output has a tag of "2" in its file name. Load this trajectory into VMD (see Exercise 0). During the trajectory, you should notice that one part of the structure is rigid and the other part is flexible.

6.5 On your own: Determine the effects of the Weld constraints and rigidification

Try holding the helices together with just base pairing forces rather than constraints. This is

a matter of removing the lines specifying the Weld constraints and rigidification. Does the domain structure change much?

6.6 On your own: Turn the RNA into a different 3D structure

Change the sequence and/or constraints and turn the RNA into a different 3D structure, e.g., a hairpin or a pseudoknot.

7 Exercise 3: Homology modeling of RNA

7.1 Objectives

MMB was the first code to do homology modeling of RNA (Flores et al., Pac. Symp. On Biocomputing 2010). In this exercise, you will:

- Learn how to use MMB to construct a model using a known RNA structure as a template (this process is known as homology modeling)
- Practice using the `AllHeavyAtomSterics` contact force
- Learn how to use `threading` forces

7.2 Specifying the template

The template is the known RNA structure.

7.2.1 Read in the PDB file for the template

You will need to read in the PDB file for this RNA (see Exercise 2). In this exercise, you will be using the *1GID.shifted.pdb* file.

In your MMB folder, you should see the following files: *1GID.shifted.pdb* and *commands.RNA-homology-modeling.dat*. If you do not see them, make sure you are in your MMB folder (see Exercise 0).

Copy *1GID.shifted.pdb* to *last.1.pdb* by typing the following in your command prompt/terminal window:

(Mac OS, Linux) `cp 1GID.shifted.pdb last.1.pdb`

Open up *commands.RNA-homology-modeling.dat* in a text editor. Verify that `firstStage` is set to `2` so that the provided PDB file is read in and used by MMB.

7.2.2 Specify template sequence to match information in the PDB file

In the input parameters file, you will also need to specify a template sequence with a chain ID and residue numbering that matches that of the PDB file. If you open the file *1GID.shifted.pdb* in a text editor, you will see that the first residue is numbered "220" and has a chain ID of "Q." So, in the command file, you would include the following line:

```
RNA Q 220 GUCCUAAGUCAACAGAUCUUCUGUUGAUAUGGAU
```

7.2.3 Rigidify the template

Lastly, you need to rigidify your template molecule so that it does not move. The threaded chain is the one that will morph so that it matches the template. In this example, the following line would rigidify the template (Tetrahymena ribozyme P6ab):

```
mobilizer Rigid Q
```

7.3 The target chain

The target chain is the one being mapped onto a known structure.

7.3.1 Specify the sequence of the target chain

In the input parameters file, specify the sequence and first residue number of the chain. For the Azoarcus fragment, this is done with the following line:

```
RNA C 146 CCUAAGGCAAACGCUAUGG
```

7.3.2 Account for sterics using "Physics where you want it"

We can use the PARM99 potential to prevent steric clashes and spread out the loop nicely. This turns on the Lennard-Jones and electrostatic terms, in addition to the bonded terms (which are on by default):

```
setDefaultMDParameters
```

Then we limit the MD forces to the *threaded* chain only:

```
includeResidues C FirstResidue LastResidue
```

Without this line, the template would also have non-bonded forces active, and would repel the threaded chain. In the original (Flores et al., RNA 2010) article, we used the `contact` command, you will see a note on this in the input file:

```
#contact AllHeavyAtomSterics C 146 164
```

This is faster, but can be a bit limited in preventing steric clashes, and won't have the long-range electrostatic repulsion that we find useful in this exercise.

7.4 Apply forces to pull the corresponding residues together

The `alignmentForces` keyword is explained in the Reference Guide, in our chapter on "Forces." Also see our chapter on homology modeling of proteins, in this Tutorial. First, specify that all subsequent `alignmentForces` commands will be performed with a prohibitive gap penalty, effectively restricting us to ungapped alignments:

```
alignmentForces noGap
```

Now we set the force constant of all the `atomSpring's`:

```
alignmentForces forceConstant 300.0
```

Lastly, let's issue the actual alignment commands:

```
alignmentForces C 146 151 Q 222 227
alignmentForces C 160 164 Q 247 251
```

In the first line above we are asking for residues 146-150 of the model ("C") to be aligned with residues 222 to 300 of the template ("Q"). For each pair of corresponding residues, this command looks for all (non-hydrogen) atoms in the first residue which have atoms with the same name in the corresponding second residue. It then applies a spring to pull those two atoms together. The spring has an adjustable force constant, which we earlier set to 300.

7.5 Run example

Make sure you are still in the directory with the *commands.RNA-homology-modeling.dat* and the *parameters.csv* files. To do this, in your command prompt/terminal window (see Exercise 0), type:

(Mac OS, Linux) `ls`

You will see a list of files in your current directory. Make sure you have *commands.RNA-homology-modeling.dat* and *parameters.csv*. If not, navigate to the directory with these files (see Exercise 0).

Now, run this example by typing:

(Mac OS) `./MMB-c commands.RNA-homology-modeling.dat`
(Linux) `./MMB -c commands.RNA-homology-modeling.dat`

The trajectory from this simulation run is in *trajectory.2.pdb* file. Load this trajectory into VMD (see Exercise 0). At the beginning of the trajectory, you should see two distinct structures. Eventually, you should see one end of the Azoarcus fragment appear to attach itself to the Tetrahymena fragment and then gradually align the rest of itself onto Tetrahymena. At the end of the trajectory, you will get a structure like that shown below, where the Tetrahymena template structure is in green and the Azoarcus target fragment is in blue. Notice that we did the homology modeling even though there are portions of Azoarcus that do not match to any parts of Tetrahymena; we dealt with this by only applying forces to corresponding bases, and leaving the rest alone.

8 Exercise 4: Protein homology modeling

8.1 Objectives

In this exercise, you will:

- Learn how to create protein chains
- Practice using the `AllHeavyAtomSterics` contact force
- Use the `threading` forces for protein chains

8.2 Specifying the template

The template is the known protein structure.

8.2.1 Provide the PDB file for the template

You will need to read in the PDB file for this RNA (see Exercise 2). In this exercise, you will be using the *protein-template.pdb* file.

In your Installation folder, you should see the following files: *protein-template.pdb* and *commands.protein-homology-modeling.dat*. If you do not see them, make sure you are in your installation folder (see Exercise 0).

Copy *protein-template.pdb* to *last.1.pdb* by typing the following in your command prompt/terminal window:

(Windows) `copy protein-template.pdb last.1.pdb`
(Mac OS, Linux) `cp protein-template.pdb last.1.pdb`

Open up *commands.protein-homology-modeling.dat* in a text editor. Verify that `firstStage` is set to `2` so that the provided PDB file is read in and used by MMB. There

are a some reporting and simulation parameters which you're by now familiar with, and we'll skip the explanation of those.

8.2.2 Start the run

(Mac OS) `./MMB—c commands.protein-homology-modeling.dat`

(Linux) `./MMB —c commands.protein-homology-modeling.dat`

Note that the last Windows release was 2.14. Windows users will therefore find that the tutorial does not exactly follow the contents of their command files. I actually hate Windows. There, I've said it! I will try to get around to another Windows release at some point, but in the meantime perhaps run the Linux version in a virtual machine.

8.2.3 Specify template sequence to match information in the PDB file

In the command file, you will need to specify a template sequence with a chain ID and residue numbering that matches that of the PDB file. If you open the file *protein-template.pdb* in a text editor, you will see that the first residue is numbered "94" and has a chain ID of "E." So, in the command file, we have the following line:

```
protein E 94 CYDYDAIPWLQNVEPNLRPKLLLKHNLFLLDNIVKPIIAFYYKPIKTLNGHEIKFIRKEEYIS
```

8.2.4 Specify model sequence, for which no structural information is available

You will also need to specify a model sequence with a chain ID (here we use "H" as a mnemonic for "human") and residue numbering which should probably follow some biological convention. We got our sequence from the telomerase database (telomerase.asu.edu):

```
protein H 522 RSPGVGCVPAAEHRLREEILAKFLHWLMSVYVVELLRSFFYVTETTFQKNRLFFYRKSVE
```

8.2.5 Rigidify the template and constrain it to ground

You will need to rigidify the template, but not leave the model flexible. You might also want to constrain the template to ground, though that's a matter of taste. Anyway, you know how to do this:

```
mobilizer Rigid E 94 156
constrainToGround E 94
```

8.2.6 Globally align the model and template backbones (with gaps)

Lastly, we will pull the model backbone into structural alignment with the template backbone based on sequence identity.

The syntax of the `alignmentForces` is in our chapter on "Forces" in the Reference Guide. In a nutshell, this is a utility that aligns sequences, and applies `atomSpring` forces between like-named atoms in corresponding residues under that alignment. The `alignmentForces` keyword admits parameters or commands. Parameters apply to commands that are below that parameter in the input file, but not to any commands that are above it. So let's set the parameters for the alignment first. Start with the force constant for the `atomSpring's`:

```
alignmentForces forceConstant 300
```

First we specify that we want to allow gaps:

```
alignmentForces gapped
```

The above simply sets a reasonable gap penalty for the global alignment. Actually it is the default setting, but it is good practice to set explicitly.

Next we tell it which chains need to be globally aligned:

```
alignmentForces H E
```

In the above we are using `alignmentForces` as a command, and passing two arguments – the chain IDs to be aligned. Global alignments have a high potential to be crappy locally, so make sure you check the alignment. This is explained in the Reference Guide but at the risk of being redundant -- search for "`SeqAn sequence alignment follows:`" in the (admittedly) verbose output. You may see something like this:

```
102
        0    .    :    .    :    .    :    .    :    .    :
```

```
        --RSPGVGCVPAAEHRLREEILAKFLHWLMSVYVVELLRSFFYVTETTFQ
                     |  ||   | |    |     |     | |    |
        CYDYDAIPWLQNVEPNLRPKLLLKHNLFLLD-NIVKPIIAFYYKPIKTLN


     50     .    :
        KNRLFFYRKSV---
             | ||
        GHEIKFIRKEEYIS
```

.. where "-" are deletions, and "|" are perfect matches.

8.2.7 Locally align based on fragments (no gaps)

If you are not happy with this gapped alignment you can specify an ungapped alignment:

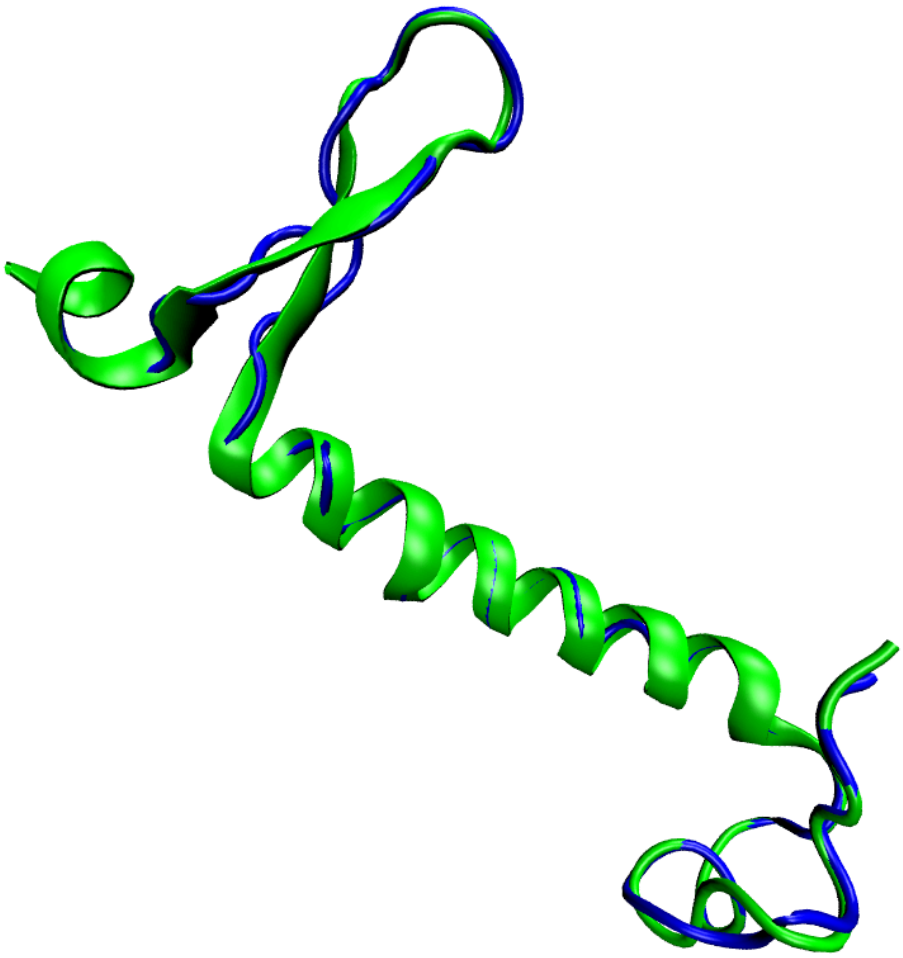
```
#alignmentForces noGap
```

This sets the gap penalty to a prohibitively high value. Then we can also specify fragments (residue ranges) to be aligned:

```
#alignmentForces  H     524     543     E       96      115
#alignmentForces  H     544     580     E       117     153
```

In the first line above we are asking for residues 524-543 of the model ("H") to be aligned with residues 96 to 115 of the template ("E"). Similarly, chain H 544-580 vs. chain E 117-153. How do we know that these residue stretches should align? From the sequence alignment (again, telomerase.asu.edu). There is a single-residue insertion in chain E -- residue 116. Correspondingly, we do not align this residue with any on chain H.

.. And we're done! The output should look something like this:



Where the template is in green and the model is in blue.

9 Exercise 5: Protein morphing

9.1 Objectives

Using what you learned earlier, you will learn to generate a putative trajectory between two known conformations of a macromolecule, a technique known as *morphing*. This will give you practice in:

- Using the `threading` command (for a slightly different purpose).
- Using the `mobilizer` command.
- Using the `readAtStage .. readBlockEnd` conditional blocks.
- Adjusting the `reportingInterval`

9.2 Introduction

We will morph Glutamine Binding Protein (GlnBP), a molecule somewhat larger than any we've worked with up to now. GlnBP is a domain hinge bending protein, meaning that it has stable structural domains connected by a flexible hinge. We will take advantage of this property by rigidifying the two domains of the model for time savings. This will get the model most of the way towards the target molecule. As an exercise, in a final stage you will leave the model flexible to complete the morph. You will see that this exercise is like the preceding homology modeling exercise, with one key difference: in morphing, not only the target's, but also the model's initial atomic coordinates are known.

Morphing is an old technique, and many good servers (e.g. molmovdb.org) and programs are available. You will see that selective rigidification offers the advantage of speed. In published work, we have morphed the entire ribosome, including all 50 protein subunits, in about 2.5 hours of computer time. Using MMB also gives you more control over precisely how the morph is done. The price of all this is that the process is bit more manual, but you will learn how to do it here.

9.3 Preparing the input structure file

As in the previous exercise, we will put out given coordinates in last.1.pdb. However this time we have coordinates for the model (1GGG.short.pdb) as well as the target (1WDN.short.pdb), and we will have to put both in the input structure file. In Mac and Linux this is easy:

(Mac OS, Linux) `cat 1GGG.short.pdb 1WDN.short.pdb > last.1.pdb`

In Windows, you will probably have to do this by cutting and pasting using your text editor.

9.4 Start the run

Start the job as usual:

(Windows) `MMB.2_14.exe –c commands.protein-morphing.dat`
(Mac OS) `./MMB –c commands.protein-morphing.dat`
(Linux) `./MMB –c commands.protein-morphing.dat`

9.5 Examine the input file

As in the previous exercise, we have model and target sequences:

```
# Model, 1GGG
protein A 5  LVVATDTAFVPFEFKQGDLYVGFDVDLWAAIAKELKLDYELKPMDFSGIIPALQ
TKNVDLALAGITITDERKKAIDFSDGYYKSGLLVMVKANNNDVKSVKDLDGKVVAVKSGTGSVDYAKA
NIKTKDLRQFPNIDNAYMELGTNRADAVLHDTPNILYFIKTAGNGQFKAVGDSLEAQQYGIAFPKGSD
ELRDKVNGALKTLRENGTYNEIYKKWFGTE
# Target, 1WDN
protein B 4 KLVVATDTAFVPFEFKQGDLYVGFDVDLWAAIAKELKLDYELKPMDFSGIIPAL
QTKNVDLALAGITITDERKKAIDFSDGYYKSGLLVMVKANNNDVKSVKDLDGKVVAVKSGTGSVDYAK
ANIKTKDLRQFPNIDNAYMELGTNRADAVLHDTPNILYFIKTAGNGQFKAVGDSLEAQQYGIAFPKGS
DELRDKVNGALKTLRENGTYNEIYKKWFGTEPKQ
```

Chain B has some residues at the N- and C-termini which don't exist on chain A. However you can verify that residue A 5 corresponds to B 5, and so on all the way to residue 224. So we will pull those residues together like this:

MORPHING

```
threading A 5 224 B 5 224
```

We will be doing this in two stages – one for rigid body alignment and another for semi-rigid morphing, as we will explain:

```
firstStage 2
lastStage 3
```

Initially we will use a `reportingInterval` of 10 ps, but you may reduce this later.

```
reportingInterval 10.0
numReportingIntervals 25
```

We will be constraining residues to ground later, so let's keep turn off the rigid body momentum removal:

```
removeRigidBodyMomentum false
```

The target molecule will be rigid throughout this exercise:

```
mobilizer Rigid B 4 227
```

9.5.1.1 Turning any parameter into a staged parameter

Any parameters or commands can be enclosed in `readAtStage ... readBlockEnd` tags. This means that the enclosed parameters will be read only for the specified stage. So if you want to read certain values for parameter1, parameter2, etc only during stages 3, do this:

```
readAtStage 3
parameter1 value1
parameter2 value2
…
readBlockEnd
```

You can have as many of these blocks as you wish, and use them to change the parameters at many stages. There are some other block markers which behave differently (e.g. `readFromStage, readToStage, readUntilStage, readExceptAtStage`), see the *Reference guide*. Also, there are a few nuances to keep track of. The input file is read from top to bottom. Parameters encountered more than once in the input file are overwritten with the one closer to the bottom of the file prevailing. Commands (e.g. `baseInteraction`), on the other hand, are additive, rather overwriting each other. See also Appendix: Forces.

Our first task is to rigidly align the model and target, since they start out spatially quite separated. We will do this at stage 2, using the `readAtStage` command just introduced. At this stage, the model will be completely rigid:

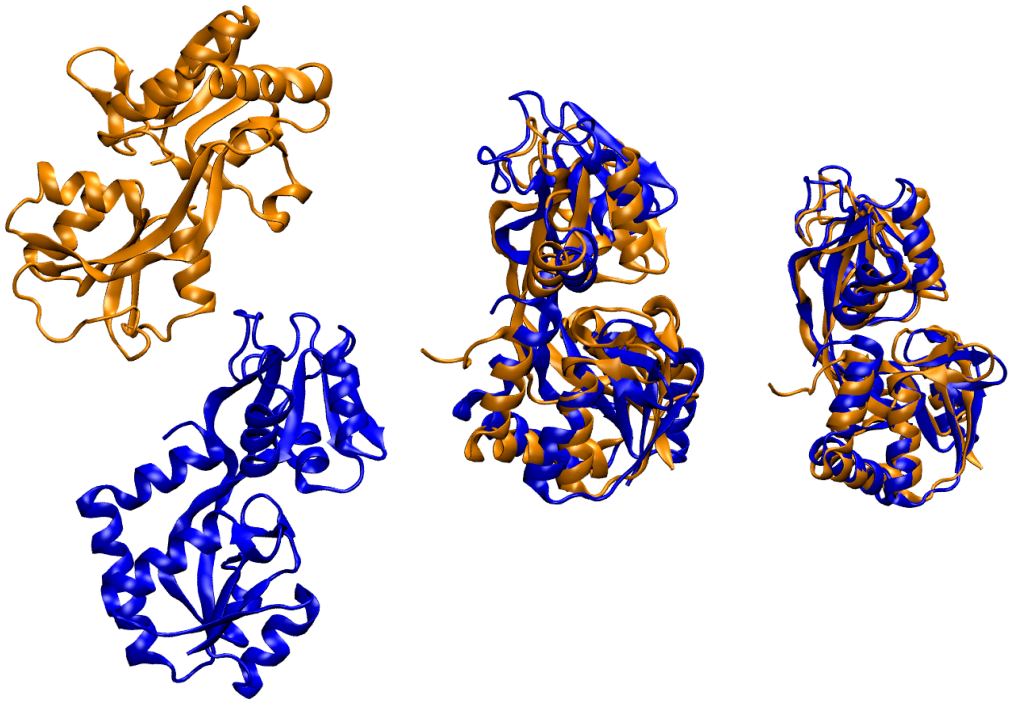
```
# Rigid alignment stage
readAtStage 2
mobilizer Rigid A 5 224
readBlockEnd
```

Then at stage 3 we will do the semiflexible morphing. We know from published work that there is a hinge at residues 88-89 and 181-183. So we rigidify the thus-defined structural domains, and constrain one of them to the ground:

```
readAtStage 3
mobilizer Rigid A 5 87
mobilizer Rigid A 90 180
mobilizer Rigid A 184 224
constrainToGround A 5
constrainToGround A 195
readBlockEnd
```

Stages 2 and 3 should be done by now. Open trajectory.2.pdb and trajectory.3.pdb in your molecular viewer. You should see something like the following:

MORPHING



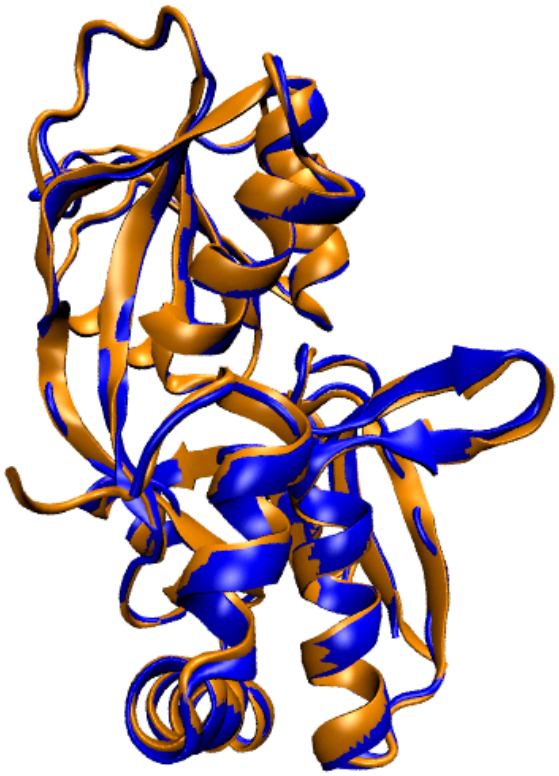
Left: Model (blue) and target (gold) in their initial, separated positions. Center: Model and target rigidly aligned. Right: Model semiflexibly aligned with target.

9.6 On your own: complete the morph with a fully-flexible alignment

You will notice that at the end of stage 3, the model is not fully aligned with the target. In this exercise, you will make the model fully flexible. You should end up with something like the image below.

Hints:

1. You will need to do this at stage 4.
2. Check that the model is fully flexible.
3. The simulation will be slower now, so reduce the reporting interval by ~ 10X.



10 Exercise 6: Template-based docking

Contributed by Eloy Vallina

10.1 Objectives

In exercise 6, you will learn how to use MMB to investigate a possible interaction between two proteins, using a technique known as template docking. In this case, we will assume that the structure of the dimer –the bound state—is known, as well as the structures of the isolated monomers. This is a good opportunity for you to use PhysicsWhereYouWantIt; in particular, the command
includeResiduesWithin <distance> <chain> <residue number>
allows you to turn on physics for all atoms in the proximity of a particular stretch of residues, within a certain distance.

10.2 Introduction

We will apply template docking to the interaction between an antigen, chicken egg-white lysozyme, and an antibody specifically selected to bind to it. The structure of the antigen-antibody complex is found in the PDB with the code `1mlc`; while the structures of chicken egg-white lysozyme can be obtained from `1dpx` and the antibody is found with the code `1mlb`. Of course, this is just a trivial example in which target and template will perfectly match, but it should give you an idea of what can be done in similar cases.

10.3 Input structures

The instructions written in `commands.template-docking.dat` rely on the following files being present in your working directory: `1mlb.AB`, `1dpx.C` and `1mlc.DEF`. These files contain the structures of the isolated antibody (2 chains), the isolated lysozyme (1 chain) and the complex (3 chains); where the chain identifiers have been altered so that they are unique during MMB run.

10.4 Start the run

(Windows) MMB.2_14.exe –c commands.template-docking.dat
(Mac OS) ./MMB.2_18.OSX –c commands.template-docking.dat
(Linux) ./MMB.2_18.Linux –c commands.template-docking.dat

10.5 Examine the input file

This MMB run starts in stage 2 so that we can read input proteins from the PDB files.

```
# BEGIN
firstStage 2
lastStage  2
```

The only staged block in this input file loads the chains with a command that is already known to you, keeping the chain identifiers as found in those files:

```
loadSequencesFromPdb 1mlb.AB.pdb
loadSequencesFromPdb 1dpx.C.pdb
loadSequencesFromPdb 1mlc.DEF.pdb
```

The next commands introduce a small displacement along the X-axis between the target antigen and antibody, so that the system is not too crowded at the beginning the run, what could cause slow performance during the first few intervals.

```
initialDisplacement A -3  0 0
initialDisplacement B -3  0 0
```

As seen before, the `alignmetForces <chain> <chain>` command pulls the ABC targets towards their DEF template counterparts, with a spring strength determined by

```
alignmentForces forceConstant  300
```

Since we want antibody and antigen to adapt to each other as they come together we will turn on the van der Waals and Coulomb terms of the force field with default parameters

```
setDefaultMDParameters
```

but we will constrain the calculations to a few amino acids in the interface by specifying a 0.60 nm radius around a few residues:

```
includeResiduesWithin .60 A 32
includeResiduesWithin .60 A 93
includeResiduesWithin .60 B 33
includeResiduesWithin .60 C 68
```

As usual, we will deactivate all physics for the template chains:

```
deactivatePhysics D
deactivatePhysics E
deactivatePhysics F
```

Additionally, we fix the template chains to the ground:

```
rootMobilizer D Weld
rootMobilizer E Weld
rootMobilizer F Weld
```

Finally, we rigidify all chains

```
mobilizer Rigid
```

and we prevent the antibody chains from getting apart from each other

```
constraint A FirstResidue Weld B FirstResidue
```

We do want to avoid clashes between chains A and C so we are going to mobilize a short stretch around the residue 45 in chain C

```
mobilizer Default C 45-1 45+1
```

and we constrain the two rigid fragments across the flexible stretch

```
constraint C 45-2 Weld C 45+2
```

10.6 On your own: docking a different antigen

The above example exploits a fortunate situation, in which the complex structure is available for exactly the two proteins participating in the interaction, as proof of principle. More often, the complex structure is missing and the purpose of template docking is to obtain information about what the interface might look like. You can try docking a lysozyme from a different source, and consider whether the same antibody would also be able to bind an antigen found in a different species.

11 Exercise 6: Efficiently generate alternate protein conformations

11.1 Objectives

In exercise 2 you learned how to use rigidification, `randomizeInitialVelocities`, and sterics to do thermal exploration of RNA conformations. In this exercise you will do the same for protein. Specifically this will teach you:

- The `ProteinBackboneSterics` sterics type arameter
- An efficient way to generate alternate conformations of proteins.
- Doing alignments and RMSD calculations in VMD

11.2 Introduction

There are many reasons to generate alternate conformations of proteins. Maybe you want to make an ensemble for protein-protein or protein – small molecule docking. Maybe you are trying to elucidate functional mechanisms. In any case, generating alternate conformations is often done by randomly moving atoms, or by using normal modes. These methods do not conserve the correct domain structure. For many hinge bending proteins, domain structure is conserved throughout the motion to some degree. In this exercise, you will find the alternate conformations that are possible under the assumption that only the hinge residues are flexible. You will use the `ProteinBackboneSterics` scheme, in which only the N, Cα, and C atoms get collision detecting spheres, to avoid generating clashing structures.

11.3 The command file

Open the file `commands.GlnBP-thermal-exploration.dat`. Most of the contents will be familiar to you. You haven't used this sterics parameter before:

```
contact ProteinBackboneSterics A 1 220
```

The hinge residues are 89-90 and 180-182:

```
mobilizer Rigid A 1 88
mobilizer Rigid A 91 179
mobilizer Rigid A 183 220
```

The first and third fragments comprise a discontinuous domain that we will fix to ground:

```
constrainToGround A 1
constrainToGround A 220
```

Leaving the second domain all the motion permitted by the hinge (and sterics).

11.4 Run MMB

Copy `1GGG.short.pdb` to `last.1.pdb`. The former is an open form of Glutamine Binding Protein (GlnBP).

Now run MMB against the command file, e.g.:

```
./MMB      –c commands.GlnBP-thermal-exploration.dat
./MMB  –c commands.GlnBP-thermal-exploration.dat
```

This will create a `trajectory.2.pdb`.

11.5 Analyze the conformational coverage

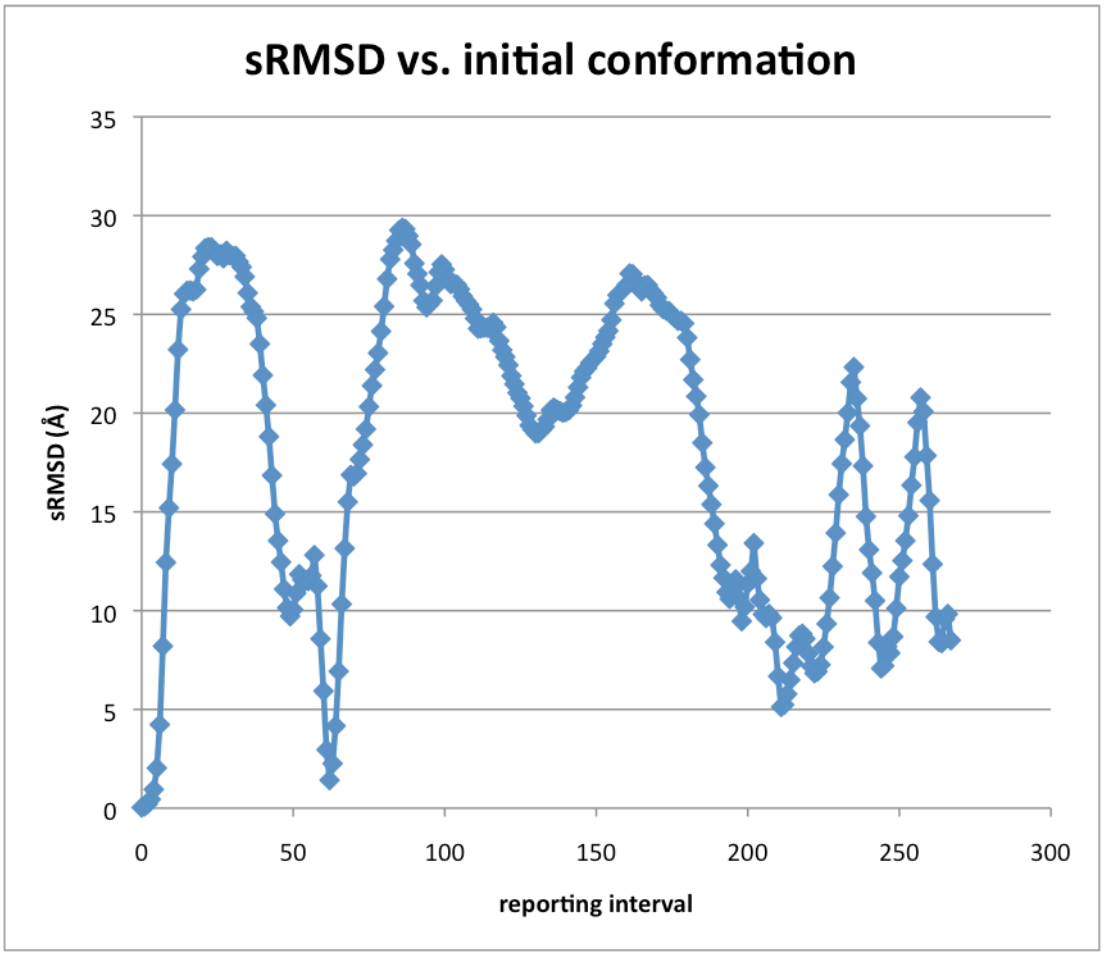
The idea behind a thermal exploration of this nature is that your ensemble may contain an alternate conformation which exists under certain conditions. You would not know this alternate conformation in a practical situation, so to have more confidence that your sampling is reasonably comprehensive, you might see how often the trajectory returns to its initial conformation, within perhaps 2Å RMSD or so. Actually we calculate this RMSD only over the mobile domain (in our case residues 87 to 175), a quantity Ruben Abagyan calls sRMSD. You can easily calculate this sRMSD using VMD. A sample script follows:

```
# loop a variable i from 1 to 269:
for {set i  1} {$i < 269} {incr i} {
# select residues 91 to 179 of the reference structure (frame 0, or the starting conformation).  “atomselect 0” means choose the first (in this case, the only) trajectory that is loaded in VMD. Put this structure in a variable called sel0:
set sel0 [atomselect 0 "resid 91 to 179" frame 0];
# create a selection set (sel1) consisting of the same domain in frame i :
set sel1 [atomselect 0  "resid 91 to 179"  frame $i];
# compute the RMSD between sel0 and sel1 :
set my_rmsd [measure rmsd $sel0 $sel1] ;
# print the RMSD :
puts $my_rmsd
# end the loop:
}
```

You can put this script in a file and read it in. It’s pretty easy just to dump it as a single line into the TK console (Extensions -> TK Console), like this:

```
for {set i  1} {$i < 269} {incr i } { set sel0 [atomselect 0 " resid 91 to 179  " frame 0 ]; set sel1 [atomselect 0  "  resid 91 to 179  " frame $i]; set my_rmsd [measure rmsd $sel0 $sel1] ; puts $my_rmsd   }
```

You will get a stream of sRMSD numbers. Cut and paste these into your favorite spreadsheet. You should be able to make a graph like this:



Note the sRMSD dips below 5Å a couple of times. You can run this longer if you are not convinced you've gotten good sampling. You can also compare this to the closed structure (PDB ID: 1WDN). That requires some aligning and careful definition of `sel0` and `sel1`.

12 ZEMu: Zone Equilibration of Mutants

12.1 Objectives

You will learn to create a *flexibility zone* about the site of a proposed mutation, locally equilibrate the mutant and wild type under an enclosing *physics zone*, and evaluate the change in binding energy. This, in a nutshell, is ZEMu. This exercise will expose you to the following parameters and command:

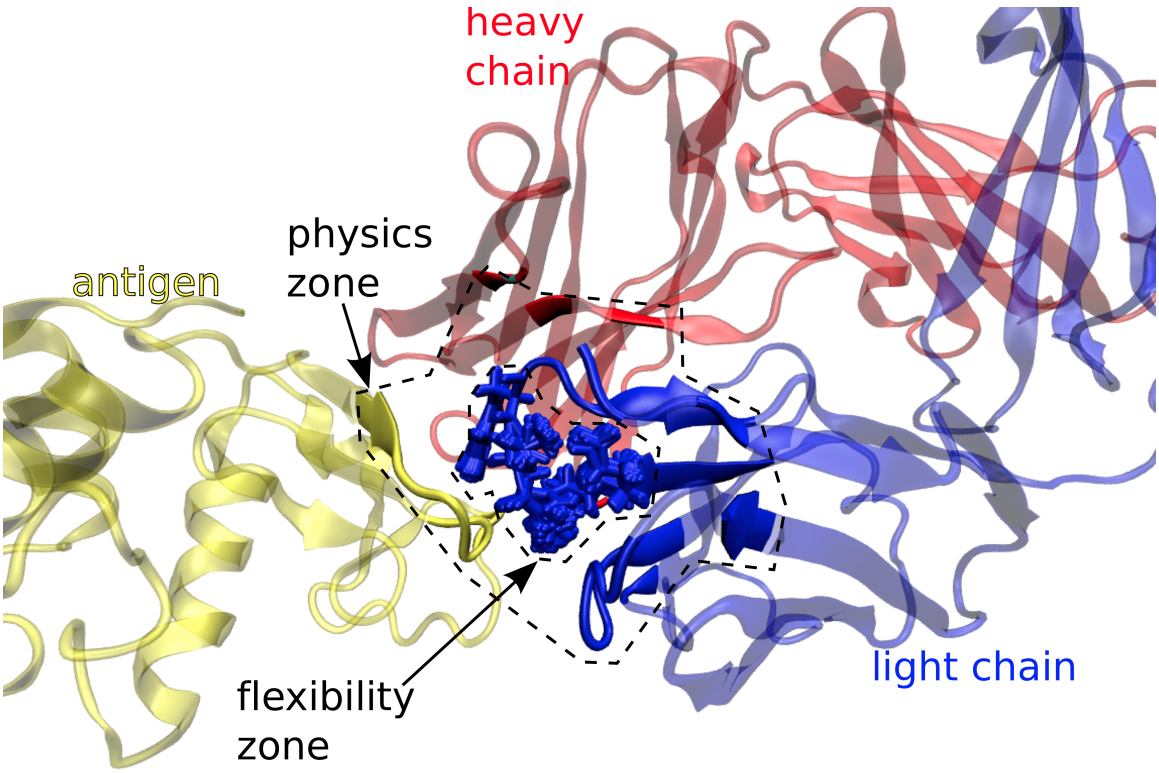
- `setDefaultMDParameters`
- `smallGroupInertiaMultiplier`
- `constrainChainRigidSegments`
- `includeAllResiduesWithin ..`

12.2 Introduction

Antibodies are powerful mediators of the immune response, and synthetic monoclonal antibodies (mAbs) can potentially be used to direct the immune system's considerable might against a pathogen. Synthetic antibodies are also used independently of the immune system, as agonists or antagonists of disease linked cell surface receptors, to deliver diagnostic markers, radioactive, or otherwise cytotoxic particles to specific locations in the patient, or in laboratory tests.

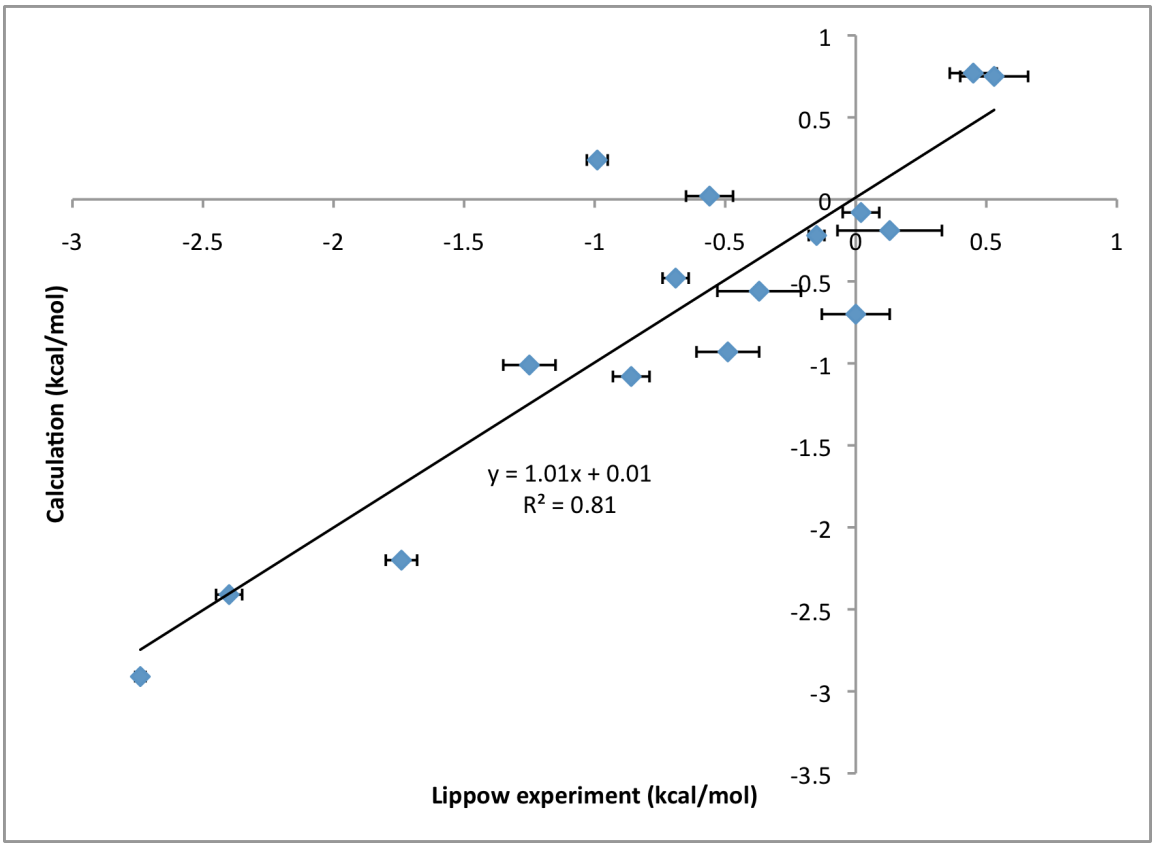
The simplest nontrivial design task is that of improving the affinity of an existing antibody-antigen complex of known structure by designing point mutations in the antibody. In order to do this one must (1) generate the point mutation and resolve the resulting steric clashes, (2) equilibrate the complex, and (3) evaluate the change in binding affinity. You can do the first two with MMB, and for the third you will need to download the FoldX program for your platform.

In this work we account for backbone and side chain flexibility employing an unusually small flexibility zone in internal coordinates. We do an equilibration of this zone taking into account physical interactions only within a small neighborhood of the flexible zone. We do this once for each possible mutant and once for the wild type complex, then use a knowledge-based potential as mentioned to evaluate the interaction energy of each, thereby obtaining the change in binding energy induced by each mutant.



***ZEMu* equilibration for mutant L:N92A.** Five residues (90-94) about the mutated light chain residue 92 form the *flexibility zone* (blurred blue licorice); their backbone and side chain bond torsions can vary while the rest of the protein residues are rigid and fixed to ground. The *physics zone* consists of residues within 9Å of the *flexibility zone* (based on Cα distances), including the *flexibility zone* itself. Residues in the *physics zone* have nonbonded (electrostatic, van der Waals, etc.) and bonded PARM99 force field terms turned on. The remainder of the protein residues are rigid, fixed, and non-interacting.

Standard error for this method is 0.4 kcal/mol. The files needed to reproduce the graph below are available from simtk.org/home/antibodydesign .



Calculated binding energy vs. (Lippow) experiment. The graph shows ΔΔG predictions from ZEMu and experimental values from Lippow, S. M., Wittrup, K. D., & Tidor, B., Nat Biotechnol. (2007). This includes single and multiple mutants. This is a small dataset, you won't get such a high correlation on a larger dataset (see Dourado and Flores, PROTEINS 2014).

12.3 Preparing the command file

Open the file `commands.A_N.32.G.dat` . Look at the first two lines. These are the antibody light chain:

```
protein A 1
DIELTQSPATLSVTPGDSVSLSCRASQSISNNLHWYQQKSHESPRLLIKYVSQSSSGIPSRFSGS
GSGTDFTLSINSVETEDFGMYFCQQSNSWPRTFGGGTKLEIKRADAAPTVSIFPPSSEQLTSGGA
```

```
SVVCFLNNFYPKDINVKWKIDGSERQNGVLNSWTDQDSKDSTYSMSSTLTLTKDEYERHNSYTCE
ATHKTSTSPIVKSFNRNEC
```

Then there's the antibody heavy chain:

```
protein B 1
QVQLQESGAEVMKPGASVKISCKATGYTFSTYWIEWVKQRPGHGLEWIGEILPGSGSTYYNEKFK
GKATFTADTSSNTAYMQLSSLTSEDSAVYYCARGDGNYGYWGQGTTLTVSSASTTPPSVFPLAPG
SAAQTNSMVTLGCLVKGYFPEPVTVTWNSGSLSSGVHTFPAVLQSDLYTLSSSVTVPSSPRPSET
VTCNVAHPASSTKVDKKIVPRDC
```

Then there's the antigen chain:

```
protein E 1
KVFGRCELAAAMKRHGLDNYRGYSLGNWVCAAKFESNFNTQATNRNTDGSTDYGILQINSRWWCN
DGRTPGSRNLCNIPCSALLSSDITASVNCAKKIVSDGNGMNAWVAWRNRCKGTDVQAWIRGCRL
```

In chain A, we want residue 32 to be a "G", whereas in the wild type it would be "N." For that, we use :

```
substituteResidue A 32 G
```

We will be constraining to ground, so we need to set:

```
removeRigidBodyMomentum FALSE
```

If it is left at `TRUE` (the default) then MMB would try to move the system so the center of mass is at the origin, and would periodically reset the overall linear and angular momentum to zero. This is not consistent with the `constrainToGround` and similar commands.
Next rigidify the whole complex:

```
mobilizer Rigid
```

Note that the `mobilizer` command is overloaded .. we could have added parameters to specify individual chains, and even stretches of residues to rigidify. But if we specify nothing, as above, we rigidify all residues in all chains.

Then we *de*-rigidify (return to `Default` flexibility) a five-residue stretch centered around the site of the mutation:

```
mobilizer Default A 32-2 32+2
```

Here we have specified both a chain and a stretch of residues. Note that since we're centering the flexibility zone around residue 32, we can be lazy and use MMB's '+' and '–' operators, rather than our noggin.
Next we turn on the non-bonded (electrostatic and van der Waals) terms in the PARM99 potential (as well as the bonded terms – but those are on by default):

```
setDefaultMDParameters
```

But we don't want to turn on physics everywhere, just in a 12Å (1.2 nm) zone about the flexible residues (30 to 34):

```
physicsRadius 1.2
```

In the absence of viscosity it's easy for small chemical groups (e.g. methyl) to spin very fast and force the time integrator to take tiny time steps. We can artificially increase the inertia (e.g. by a factor of 11.0) of such groups to prevent this from happening:

```
smallGroupInertiaMultiplier 11
```

We also want to fix all the rigid segments of the complex to ground:

```
constrainChainRigidSegments
```

The remaining parameters should be familiar, or can be looked up in the reference guide:

```
firstStage 2
```

```
lastStage 2
reportingInterval .1
numReportingIntervals 150
```

I have provided the antibody-antigen complex. First do `cp antibody-antigen.pdb last.1.pdb .`

Now run MMB with this command file (`commands.A_N.32.G.dat`). When you're done move last.2.pdb to `A_N.32.G.pdb .`

Next run MMB using `commands.A_N.32.N.dat.` This is the wild type complex. Alternatively, just modify `commands.A_N.32.G.dat` so that the 32nd residue is "N" rather than "G".

When you're done move last.2.pdb to `A_N.32.N.pdb .`

12.4 Evaluating binding energy using FoldX

We can't distribute FoldX with MMB, unfortunately. Download that from foldx.crg.es. Then create a file called `batch.txt` containing the following two lines:

```
A_N.32.N.pdb
A_N.32.G.pdb
```

Save and close it. You find a file called `runfile.txt` in your MMB distribution. Run FoldX, choose option 3 (run file) and give it the file name "`runfile.txt`". This will evaluate the binding energy for the two equilibrated structures listed. When it's done, evaluate the difference :
(energy of `A_N.32.G.pdb`) - (energy of `A_N.32.N.pdb`)

This will give you the change in binding energy due to the mutation. Compare to the experimental value of -0.86 kcal/mol. It shouldn't be far off.

12.5 Challenge: find mutations that lower the binding energy even further

Try to find mutations that are even better than those reported. This will not be trivial!

13 Exercise 7: Solve protein structure by NMR constraints

13.1 Objectives

You've learned a lot so far. You've learned how to do everyday modeling tasks such as morphing, conformational sampling, and homology modeling. You learned how to turn base pairing contacts into 3D structure of RNA. I will progressively make things more challenging for you. In this chapter you will learn how to turn distance constraints, such as can be obtained from NMR experiments, into 3D structure with a little help from the Amber99 force field. This will involve the following tasks:

- Use the `atomTether` command
- Turn a list of distance constraints into MMB commands
- Turn on the Amber99 force field for the entire system

13.2 Instructions

You are reasonably far along now, so you don't need detailed instructions for everything – thus I'll skip a few basic steps. You will need to turn on all terms of the force field:

```
globalAmberImproperTorsionScaleFactor     1
globalBondBendScaleFactor                 1
globalBondStretchScaleFactor              1
globalBondTorsionScaleFactor              1
```

```
globalCoulombScaleFactor                    1
globalVdwScaleFactor                        1
```

In this exercise we are turning on physics everywhere. So you can just leave `physicsWhereYouWantIt` at the default value, or set it explicitly:

```
physicsWhereYouWantIt FALSE
```

You might want to use a temperature that leads to some oscillation about equilibrium:

```
temperature 100
```

You will also need the sequence. You can extract it from `1UAO.short.pdb` using the `extract_FASTA.awk` script.

Lastly, you will want to add the distance constraints. Let's say you know that on chain A, atom 2HA of residue 1 is at most .45nm from atom HE3 of residue 9. The way you would enforce that is:

```
atomTether A    1  2HA  A    9  HE3 .4500  300.00
```

where the last (optional, defaults to 30 if left out) number specifies the spring constant of a spring that will pull the two atoms together if they're more than 4.5Å apart. I suggest making this 300 for this application, because empirically I've found this is strong enough.

Unfortunately the people that made the 1UAO structure didn't use the PDB atom naming convention. So we will have to correct the atom names. The following substitutions are necessary:

Everywhere:
HA2 2HA
HB2 2HB
HG2 2HG

HG21 1HG2
HG22 2HG2
HG23 3HG2

Residue 4 only:
HD2 2HD

If you want to skip this hassle, just use `1UAO.atoms-renamed.pdb`.

I've included a list of distance constraints called `1UAO-disre-simple.txt` . Try to use it to generate the `atomTether` commands. You may find the `parse-restraints.pl` script useful.

The `parse-restraints.pl` script looks like this:

```
# perl ./parse-restraints.pl 1UAO-disre-simple.txt 1UAO.short.pdb
#open the restraints file (first argument):
open RESTRAINTS, $ARGV[0]    or die $!;
#load restraints into an array:
@restraints = <RESTRAINTS> ;
close (RESTRAINTS);
int r;
#for each restraint:
for ($r = 0; $r < scalar(@restraints); $r++)
{
    #first atom number
    int $i ; $i =  substr($restraints[$r],0,3);
    #second atom number
    int $j ; $j =  substr($restraints[$r],10,3);
    #distance:
    int $dist ; $dist =  substr($restraints[$r],20,5);
    # initialize to "*" so we can later tell if not read
    $atomName1 = "*";
    $atomName2 = "*";
    int $residueNumber1; $residueNumber1 = -1;
```

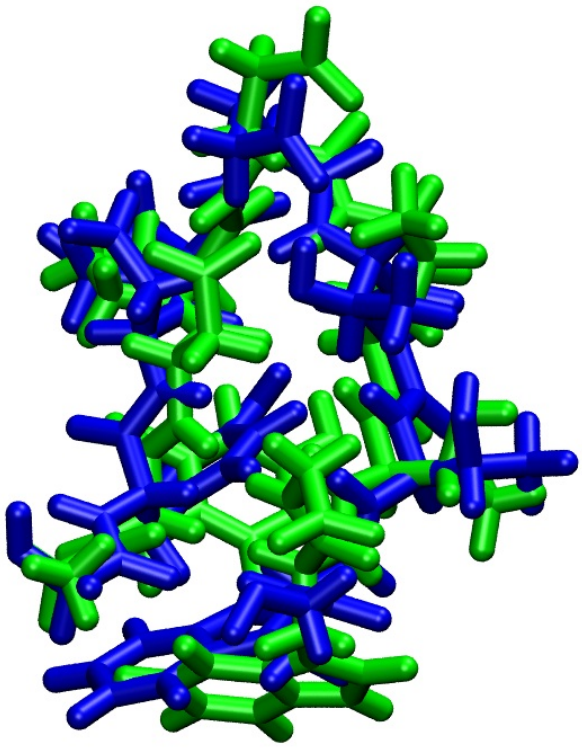
```
    int $residueNumber2; $residueNumber2 = -1;
    $chain1 = "A";
    $chain2 = "A";
    #open PDB file (second argument)
    open PDB, $ARGV[1]    or die $!;
    #for each line in PDB file:
    while (<PDB>) {
        #parse the atom number
        int $PDBatomNumber; $PDBatomNumber = substr($_,6,5);
        #if first atom number matches an atom number in the PDB file
        if ($PDBatomNumber == $i)        {
            # extract atom name, residue number, and chain ID:
            $atomName1      = substr($_,12,4);
            $residueNumber1 = substr($_,22,4);
            $chain1         = substr($_,21,1);
        }
        if ($PDBatomNumber == $j)        {
            # extract atom name, residue number, and chain ID:
            $atomName2      = substr($_,12,4);
            $residueNumber2 = substr($_,22,4);
            $chain2         = substr($_,21,1);
        }

    }
#print out MMB atomTether commands:
print  "atomTether  $chain1  $residueNumber1  $atomName1    $chain2
$residueNumber2 $atomName
2 $dist \n";
}
#done!
```

You will also need to convert the distances from nm to Å. While you're add it, set the spring constant to 300.0.

It's best to make your own MMB input file. But if you just want the right answer, look at the provided `commands.NMR.dat` .

13.3 Results

Your structure should agree with the published one (`1UAO.short.pdb` in your MMB distribution) within about 1.3Å RMSD.



maps

14 Exercise 8: Fitting to electron density maps

14.1 Objectives

You may have found some of the preceding exercises redundant in some sense, perhaps repeating in protein what was already done in RNA, etc. Perhaps you could be forgiven for falling asleep. In this exercise we will do something completely different – fitting atomic coordinates to electronic density maps, which could have come from a cryo-electron microscopy (CryoEM), small-angle X-ray scattering, crystallographic, or other experiment. The skills you picked up in previous lessons about selective rigidification, constraints, forces, even "Physics where you want it" or straight-out all-atoms force fields will serve you well as you efficiently build 3D models. The following parameters will be new to you:

- `densityForceConstant`
- `densityFileName`

The following command will also be new:

- `fitToDensity`

If you want a challenge, you can also learn how to extract the sequence (in single-letter code) from and to renumber the residues in a PDB file. Hopefully you will also gain some insight into the flexibility of the ribosome.

14.2 Introduction

Electron density maps can be produced by cryo-electron microscopy (Cryo-EM), Small Angle X-ray Scattering (SAXS), X-ray crystallography, and other means. They are an important source of structural information. However they are hard to interpret without solving for the nuclear positions. Most of the structures in the PDB were once electronic densities, and have been fitted with nuclear positions.

There are many fine pieces of software available for fitting 3D structure to density maps. At Uppsala, "O" is quite popular. I will not attempt a full review of such programs here. Our approach, however, follows the work of Klaus Schulten, who invented Molecular Dynamics Flexible Fitting (MDFF). In MDFF, the atoms in the molecule or complex are subject to a conventional Molecular Dynamics force field, plus an additional force which is proportional to the atomic mass and the gradient of the electronic density. In MMB, we adapt this force as follows:

$$\vec{f}_i = A \cdot m_i \cdot \vec{\nabla} D(x_i, y_i, z_i)$$

Where i is the atom index, m_i is the mass of atom i, $D(x_i, y_i, z_i)$ is the electronic density at the nuclear position of atom i, A is a user-adjusted scaling factor, and $\vec{\nabla}$ is the gradient operator. Accordingly, $\vec{f}_i$ is the density-derived force vector applied to atom i. This is computed for and applied to every atom i in the system.

MMB reads the .xplor density map format. It supports non-orthogonal unit cell axes, that is to say triclinic unit cells.

In this exercise, you will specify the sequence of a tRNA molecule, read in an initial set of nuclear coordinates, read in the density map of the ribosomal hybrid state, and then fit the tRNA molecule into the density. So let's get started!

14.3 Run MMB

In a practical situation, preparing a good starting model is an important part of the work. I used Venki Ramakrishnan's structure of the *T.thermophilus* ribosome in the classical state (2J00, 2J01, 2J02, 2J03), which I then semiflexibly morphed to match Jamie Cate's "R2" intermediate structure. You can read all about the why and wherefore in my 2011 paper in *Proceedings of the Pacific Symposium on Biocomputing*. Anyway, I took the morphed

structure and re-centered it using COLORES, which is part of the Situs package. This is easier than it might sound, but you won't have to do any of it, just use the coordinates in `tRNA.pdb`, which is in your MMB 2.4 distribution. Issue:

```
cp tRNA.pdb last.1.pdb.
```

Unfortunately this will actually take some time to converge. So start it now, so at least it will run for a couple of minutes while we finish going through the input file. Issue:

```
./MMB      –c commands.tRNA-fitting.dat
./MMB  –c commands.tRNA-fitting.dat
```

Depending on your OS. Note that MMB 2.4.1 has a density fitting algorithm that is a full 10X faster than MMB 2.4! So make sure you are using at least MMB 2.4.1 for this exercise.

14.4 The command file

We first instantiate a tRNA molecule:

```
RNA V 5 CGCGGGAUGGAGCAGCCUGGUAGCUCGUCGGGCUCAUAACCCGAAGGUCGUCGGUCAAAUCCGGCCCCCGCAA
```

If you don't have the `commands.tRNA-fitting.dat` file, you can extract the sequence from a structure file that contains only the tRNA, using e.g. `awk –f extract-FASTA.awk <PDB file>`. You will find `extract-FASTA.awk` in your 2.4.1 distribution.

Next we rigidify the tRNA fully:

```
mobilizer Rigid V 5 77
```

We have to turn off the rigid body momentum remover, since this would always be trying to recenter the molecules:

```
removeRigidBodyMomentum false
```

As you recall from our definition of $\vec{f}_i$ above, the scaling factor A is user-adjustable. In the command file, A is called `densityForceConstant.` Make this factor too small, and the fitting will take forever. Make it too big, and the molecule might fly out into deep space. Turns out it's probably best to leave it at the default value of unity:

```
densityForceConstant  1.00
```

Now we specify the name of the electron density file, which has to be in XPLOR format:

```
densityFileName       ./tRNA.xplor
```

Then we activate the density-based force field for chain V:

```
fitToDensity     V
```

Note that we could just as easily have issued:

```
fitToDensity     V     FirstResidue     LastResidue
```

(which does exactly the same thing), or:

```
fitToDensity
```

(which fits all chains in the system, which in this case is also the same thing)

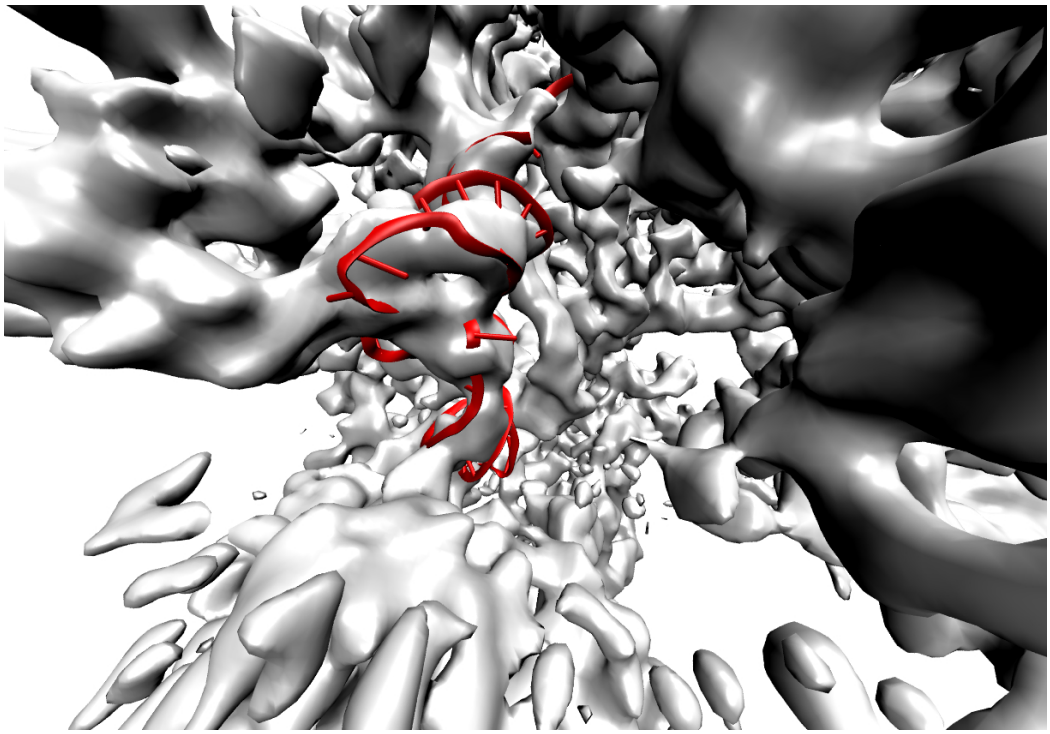
..I just wanted to make sure you understand the polymorphism of this command.

The rest of the parameters will be familiar to you.

```
temperature 1
numReportingIntervals 100
reportingInterval .01
firstStage 2
lastStage 2
```

14.5 View the results

VMD can display density maps. So read in `tRNA.xplor`. I rendered this using "Solid surface." Then read trajectory.2.pdb as a new molecule. You should be able to watch the tRNA move into its corresponding density. It should look something like this:



14.6 On your own

We just fitted the P/E site tRNA into the tRNA density map. If you want to fit a bigger subunit, try 16S. You can download the emd_1315 density map and fit the 16S from 2AVY (provided, or get from the PDB). You will need to extract the sequence of this subunit, and make everything but the neck region `Rigid`. You may consider the neck region to consist of residues 903 and 1373. Make sure you `Weld` together fragments of any discontinuous domain.

complex

15 Virtual assembly of a protein-DNA complex

Contributed by Erik Marklund

15.1 Objectives

Quite often a good experimental structure model for the particular biomolecule that you are to study is not available under the specific conditions that you demand. For instance, a protein may have been crystallized with the “wrong” ligand, lacking one or a few domains, etc. In such cases the combined data from several experiments can still be used to create a reasonable structure model that can e.g. be used for subsequent molecular dynamics. In this exercise you will see an example of how an existing protein structure model can be used in conjunction with sequence data to produce a model of a related protein with maintained protein-ligand interactions. There are no new commands in this exercise, but the alignment will go beyond the ordinary use of `proteinThreading`.

15.2Introduction

The tetracycline repressor (TetR) regulates the genes for tetracycline resistance in bacteria. It is a commonly used system for conditional gene expression and has a high affinity for its operator, *tetO*. There is a X-ray crystallographic structure of operator-bound TetR class D (TetRD) in the protein databank (id. 1QPI), but not for the related TetR class B (TetRB). Their high level of sequence similarity, however, allows for structural alignment of TetRB onto TetRD to yield a structure model of TetRB. Because of the specific interaction with DNA the side-chain conformations of the DNA-binding regions require special attention.

15.3Run MMB

The input structure file (1QPI) requires little preparation. It contains one monomer from a homodimer and one strand from a double stranded DNA helix. The crystallographic symmetry found in the pdb file can be used at a later point to generate the homodimer bound to the

double stranded DNA helix. Because of that we will only perform a structural alignment of a monomer here.

Use the structure model of TetRD as an input file:

```
cp TetR.pdb last.1.pdb.
```

Then execute MMB to do the actual alignment:

```
./MMB      –c commands.TetR_threading_TUT.dat
./MMB  –c commands.TetR_threading_TUT.dat
```

(depending on your OS). This will thread the TetRB polypeptide chain onto the TetRD structure while maintaining the side chain interactions with DNA.

15.4 The command file

First we set up the environment and instantiate the TetRD and TetRB monomers and the DNA:

```
firstStage 2
lastStage  2
reportingInterval 1.0
numReportingIntervals  50
temperature 1.0
removeRigidBodyMomentum false

# TetR class D, bound to DNA
protein A 4 LNRESVIDAALELLNETGIDGLTTRKLAQKLGIEQPTLYWHVKNKRALLDALAVEILARHHDYSLPAA...
# TetR class B, no structure
protein B 4 LDKSKVINSALELLNEVGIEGLTTRKLAQKLGVEQPTLYWHVKNKRALLDALAVEILARHKDYSLPAA...

# DNA
RNA M 1 CCUAUCAAUGAUAGA
RNA N 1 UCUAUCAUUGAUAGG
```

Perhaps you notice the high sequence similarity of the regions shown above. The structure of TetRD has some stretches of residues that were not resolved in the experiment. The TetRB sequence contains the corresponding residues and has additional insertions that will be part of the final structure model. This means, however, that care must be taken to align the right parts of TetRB to TetRD, as there is not a one-to-one mapping of all residues in the two sequences. The DNA molecules (here instantiated as RNA for technical reasons) are not necessary for the alignment, but make the final output more comprehensive.

Let's set up the homology modeling of the backbone:

```
threading A      4       155     B       4       155 300.0
threading A      156     198     B       169     211 300.0
```

Here the insertions create a discrepancy between the two sequences in terms of residue numbering, as discussed previously. The same thing affects the mobilizers that keep most of the proteins rigid throughout the homology modeling:

```
mobilizer Rigid A 4 198
mobilizer Rigid B 4 22
mobilizer Rigid B 30 34
mobilizer Rigid B 50 155
mobilizer Rigid B 169 211
mobilizer Rigid M 1 15
mobilizer Rigid N 1 15
```

The mobilizers above are further complicated by the fact that sidechains that make DNA interactions can not be kept rigid, or their final conformations will be off with respect to the DNA. Therefore we have split up one rigid part into several shorter ones.

We anchor TetRD and the DNA to the ground:

```
constrainToGround A 4
constrainToGround M 1
constrainToGround N 1
```

15.5 View the results

Fire up your molecular viewer of choice to inspect your new structure model. last.2.pdb only contains the monomeric protein and single stranded DNA. Hence you will need to make use of the crystallographic symmetry information that is contained in the input structure file.

15.5.1 Symmetry expansion with PyMOL

Copy the `CRYST1` record from last.1.pdb and the coordinates from last.2.pdb to a new file:

```
grep CRYST last.1.pdb > TetRB_threaded.pdb
cat last.2.pdb >> TetRB_threaded.pdb
```

In PyMOL you can now make a symmetry expansion. Open PyMOL, load the file `TetRB_threaded.pdb`, and execute `symexp`:

```
Load TetRB_threaded.pdb
symexp S_, TetRB_threaded, all, 1.5
```

This generates symmetry related copies of the monomeric protein and DNA locally. Note that this command is likely to create more copies than you need, so a few newly generated objects may need to be deleted from the selections/objects panel to the right. Once you have the homodimer you will be able to see if the inserted loops cause any clashes between the monomers that may need further processing. As you will see, the inserted loops are nicely situated in regions that are not occupied by any other atoms, so the entire structure is a plausible structure model of the TetRB-operator complex

The protein-DNA interface of a structurally aligned TetRB homodimer. The homodimer was constructed from the monomeric protein and DNA with the help of the `symexp` command in PyMOL. Not only is the structure model devoid of side-chain clashes; the specific interactions with DNA were reconstituted in the homology modeling process.

complex

15.5.2 Symmetry expansion using other tools

Unfortunately, VMD currently lacks the capability to create the full homodimer directly from the crystallographic symmetry information contained in the pdb file. There are other tools at our disposal, however. Examples of such are XPAND (http://xray.bmc.uu.se/usf/) and CCP4 (http://www.ccp4.ac.uk/), both of which are free to use. Unfortunately, neither XPAND nor CCP4 are guaranteed to work out of the box, but if either of them is already present on your system you could try to make use of it. Finally, there is a web service – Quat (http://sysimm.ifrec.osaka-u.ac.jp/pdb_quat/) – that can do expansions according to both crystallographic and non-crystallographic symmetry. Before submitting your structure to Quat it is strongly recommended that you remove TetRD from the pdb file! Quat may destroy the chain labeling, so it's better to have as few chains as possible before submitting it. For this reason you may choose to also omit the DNA from the Quat input file since it is already symmetry expanded. Inspect the structure afterwards to make sure that the symmetry expansion produced sensible copies!

In principle the symmetry operations can be done in VMD with the help of rotations and translations, but requires some level of familiarity with the crystallographic space groups. In this case the other monomer(s) can be generated by rotating all atoms 180 degrees around the y-axis followed by translation by half a unit cell along the z-axis. This can also be accomplished by putting your favorite scripting language to good use.

16 Graphical User Interface with Chimera

We developed a plugin for Chimera to setup and run MMB simulations from a Graphical User Interface (GUI).

16.1 Manual install of the plugin for Chimera

The first step is to install MMB as indicated in Chapter 2 of this tutorial. Then a few instructions depending on your operating system must be followed to integrate the plugin into your Chimera installation.

16.1.1 Manual install on Mac OSX

Download Chimera for your operating system and follow the install instructions.
http://www.cgl.ucsf.edu/chimera/download.html

Now look inside your install directory of MMB (see chapter 2). You should find a directory called *lib*, containing the necessary libraries to run MMB, another called *share*, containing the Python files composing the plugin for Chimera.
To allow Chimera to run MMB, you have to create symbolic links of the libraries into Chimera's own library directory using the command line.
If you installed MMB in your home directory and Chimera in /Applications, it would look as follows:

```
cd /Applications/Chimera.app/Contents/Resources/lib/
sudo ln –s ~/Installers.<MMB_version>/lib/*.dylib .
```

Now, open Chimera. Open the Tools Preferences (Favorites->Add to Favorites/Toolbar). Click on the Add… button and navigate to `/Users/<user_name>/Installer.<MMB_version>`
Select the *share* directory and click Open. Quit and restart Chimera, you should have an MMB icon on Chimera's toolbar. You can also access the plugin from Tools->MMB->MMB User Interface.

16.1.2 Manual install on Linux

Download Chimera for your operating system and follow the install instructions.
http://www.cgl.ucsf.edu/chimera/download.html

Now look inside your install directory of MMB (see chapter 2). You should find a directory called *lib*, containing the necessary libraries to run MMB, another called *share*, containing the Python files composing the plugin for Chimera.
To allow Chimera to run MMB, you have to create symbolic links of the libraries into Chimera's own library directory using the command line.
If you installed MMB in your home directory and Chimera in the default drectory, it would look as follow:

```
cd .local/UCSF_Chimera/lib/
sudo ln –s ~/Installers.<MMB_version>/lib/*.dylib .
```

Now, open Chimera. Open the Tools Preferences (Favorites->Add to Favorites/Toolbar). Click on the Add… button and navigate to `/Users/<user_name>/Installers.<MMB_version>`
Select the *share* directory and click Open. Quit and restart MMB, you should have an MMB icon on Chimera's toolbar. You can also access the plugin from Tools->MMB->MMB User Interface.

16.1.3 Manual Install on Windows

Download Chimera for your operating system and follow the install instructions.
http://www.cgl.ucsf.edu/chimera/download.html
Now, open Chimera. Open the Tools Preferences (Favorites->Add to Favorites/Toolbar). Click on the Add… button and navigate to where you installed MMB.
Select the *share* directory and click Open. Quit and restart MMB, you should have an MMB icon on Chimera's toolbar. You can also access the plugin from Tools->MMB->MMB User Interface.

16.2 Using the GUI

Open Chimera and click on MMB's icon in the toolbar (or in the menu: Tools->MMB-> MMB User Interface).

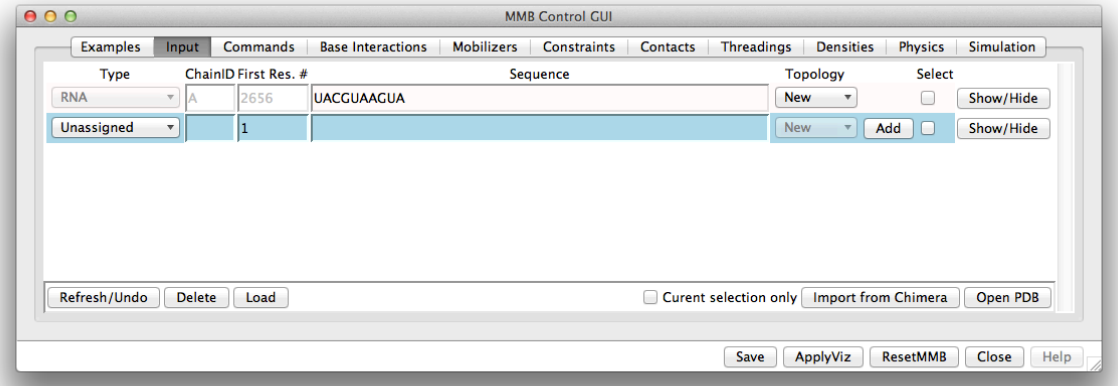
A window named MMB Control GUI appears, composed of several tabs. You should be on the *Input* tab.

WARNING: On OSX Mavericks, Chimera is unstable when using an external monitor.

16.2.1 RNA folding example

Use the dropdown lists and text field to add a new RNA chain A, starting from residue 2656 with the sequence UACGUAAGUA.

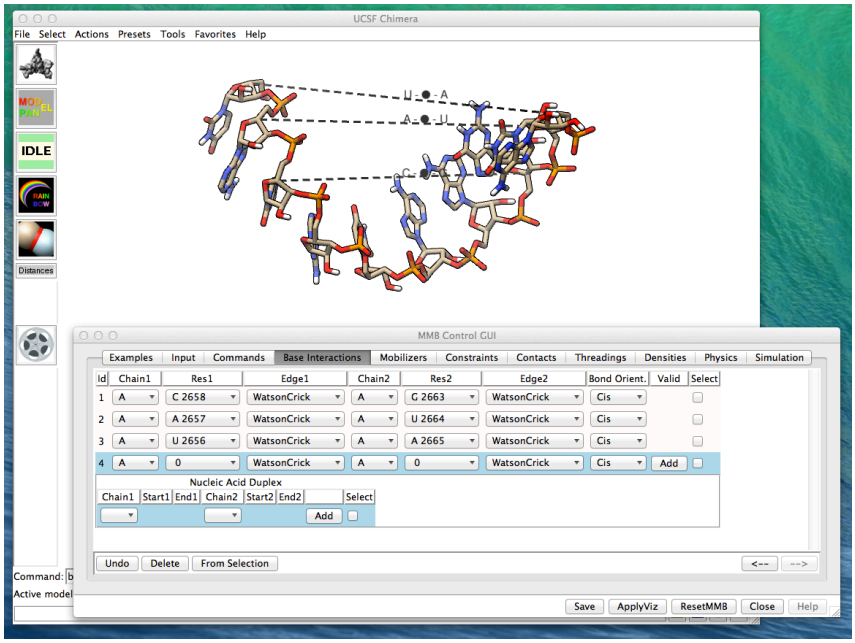
After clicking on *Add*, the window should like the following:



Now click on the *Load* button to initialize the structure. MMB generates a new strand RNA and Chimera displays it.

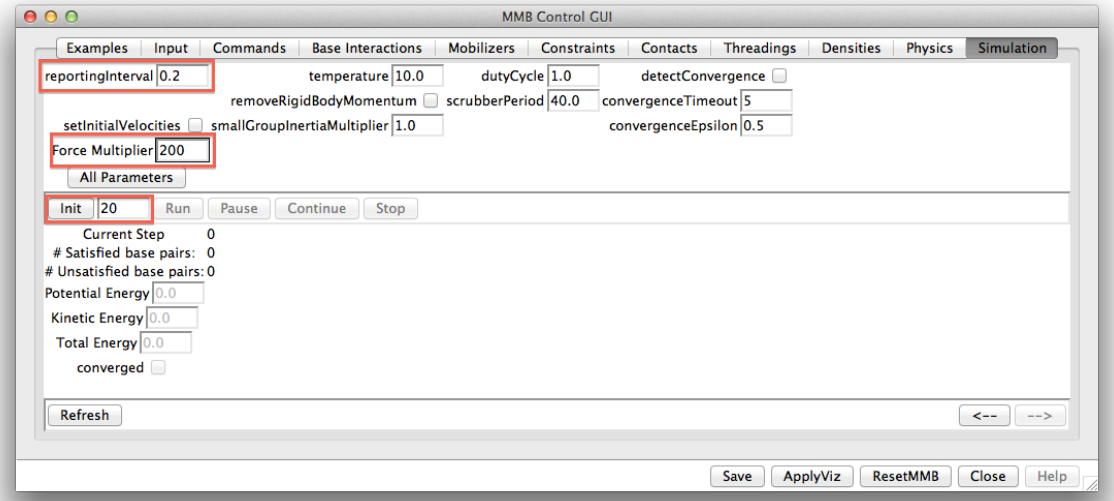
Now switch to the *Base Interactions* tab. Here you can define how the bases should interact with each other. Add three interactions corresponding to the following MMB commands:

```
baseInteraction A 2658 WatsonCrick A 2663 WatsonCrick Cis
baseInteraction A 2657 WatsonCrick A 2664 WatsonCrick Cis
baseInteraction A 2656 WatsonCrick A 2665 WatsonCrick Cis
```

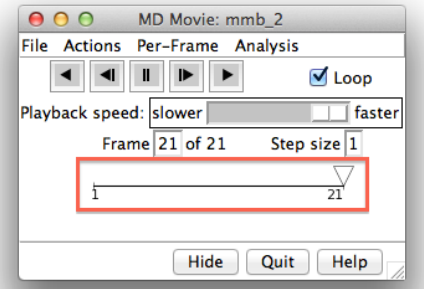


After adding these, your screen should look like the picture. On the 3D window, you can follow the change you make. Bases interactions are represented as dashed lines labeled with the kind of Leontis&Westhof interaction chosen.
If you have problem visualizing the lines, we recommend to set the background to white (Favorites->Preferences or the Chimera command: `background solid white`).

Now let's simulate the folding. Go to the *Physics* tab and activate the use of the forcefield by clicking on the *Default MD Parameters* button.
Switch to the *Simulation* tab. Set the value of *reportingInterval* to 0.2 to have a quick feedback and increase the *Force Multiplier* to 200 to enforce the base interactions.



Now you can click on the *Init* button to initialize the simulation. A window titled *MD Movie: mmb* appears. It will be used to handle the trajectory generated during the simulation.
In the main window, click on *Run* to launch the computation. The textbox on the right of the button is the number of Reporting Intervals you want.
While the simulation is running, you can see the structure changing in the 3D view. You can use the appropriate buttons to Pause/Continue and Stop the current run.
Once the run is finished, you can use the "player" in the MD Movie window to review the trajectory.



You can use this window to save the trajectory too: File->Save PDB...
Clicking on *Run* again will continue the simulation from the last computed structure. You can change the number of Reporting Intervals and id you want to automatically detect convergence.

At any time, you can click on the *Save* button to save all the commands and modified parameters into a file readable by MMB.

You can now reset MMB by clicking on the *ResetMMB* button. A dialog box asks you wether you want to reuse the current structure and parameters or to restart from scratch.
Let's restart from scratch to continue the tutorial.

16.2.2 Flexible Protein morphing