frodaN

Geometrical simulations of biomolecules
Version 1.0

USER GUIDE

1. Introduction

This user guide describes the installation and usage of frodaN version 1.0. frodaN has been developed in Michael Thorpe's group at Arizona State University.

frodaN is software for geometric simulation of biomolecules such as proteins, DNA, and RNA. Geometric simulation is an approach to molecular simulation that is very different from traditional molecular dynamics methods. It follows the philosophy that essential features of conformational flexibility and motion in biomolecules can be captured by solely considering geometric relationships between atoms. The approach has evolved over the years. A method and software package called FIRST was originally introduced in Jacobs et al. for predicting rigid and flexible regions in static protein structures. The term "geometric simulation" was first coined (as far as we are aware) in Wells et al., which introduced a method known as FRODA (Framework Rigidity Optimized Dynamics Algorithm), for exploring conformational space of a protein subject to geometric constraints.

Since the publication of the FRODA paper, a new geometric simulation method has been developed and published in Farrell et al., similar in its overall concept to FRODA but featuring the following improvements: (1) a completely redesigned geometric model and mathematical framework for enforcement of constraints that converges more robustly, more tightly, and more quickly; (2) very large sampling speed ups from a new exploration strategy called momentum run-on; (3) redesigned geometric constraints including constraints that keep backbone dihedrals out of disallowed Ramachandran regions, constraints that prevent eclipsed side chain torsion angles, and in particular, non-overlap constraints that have been calibrated to Amber (to be published); (4) new "geometric targeting" capability for rapid generation of stereochemically-acceptable all-atom pathways in proteins and other biological molecules.

This software, frodaN, implements the new geometric simulation/targeting method described in Farrell et al. The "N" in frodaN stands for "New", to reflect the significant changes to the model and methodology, and to emphasize that this is a new software package written from scratch, rather than an adjustment to the original FIRST/FRODA software. Also, the "froda" in "frodaN" is no longer an acronym, but a name that reflects the parentage of the method.

This frodaN software also underlies two easy-to-use webservers for those that prefer a web-interface to a command-line: (1) the Flexweb webserver http://flexweb.asu.edu runs frodaN to explore conformational space of a biomolecule subject to geometric constraints, similar to FRODA; (2) the new Geometric Pathways webserver http://pathways.asu.edu runs frodaN to generate all-atom pathways between two given states of a biomolecule.

This command-line version of frodaN can be run in two ways: (1) to generate a pathway between two given structures subject to geometric constraints; (2) to explore conformational space subject to geometric constraints, starting from a given structure.

The software is a work in progress; it works well provided the input file is formatted properly and does not contain erroneous data. There is a limited amount of error checking that occurs. Bugs can be emailed to asucbp@gmail.com. The main focus of this project is on the development of the frodaN software, with a lesser emphasis on applications. As such, we would appreciate any feedback on the software, as it will help guide future directions for development. Finally, we will be available as much as possible for support, and hope you enjoy using the software!

Jul 22, 2010
Biophysics Theory Group (Professor Michael Thorpe)
Arizona State University
Departments of Physics and Chemistry & Biochemistry

1.1. Citing frodaN

Please cite:

Daniel W. Farrell, Kirill Speranskiy, M.F. Thorpe
Generating Stereochemically Acceptable Protein Pathways
Proteins: Structure, Function, and Bioinformatics (In Press, 2010)
doi: 10.1002/prot.22810

2. Third Party Software

frodaN is written in C++ and Python, and includes the following software/source code:

FIRST
Floppy Inclusions and Rigid Substructure Topography
http://flexweb.asu.edu

Hybrid-36
http://cci.lbl.gov/hybrid_36/

SeqAn
Library for Sequence Analysis
http://www.seqan.de/dddoc/html/index.html

TCLAP 1.2.0
Templatized C++ Command Line Parser Library
http://tclap.sourceforge.net/

TinyXML++
http://code.google.com/p/ticpp/

Mersenne Twister (MT19937)
http://www.math.sci.hiroshima-u.ac.jp/~m-mat/MT/emt.html

Timer.h
A CPU timer C++ class from the software package "dch"
https://simtk.org/home/dch

Vect.h
A 3-D vector C++ class from the software package "dch"
https://simtk.org/home/dch

libtricubic
http://orca.princeton.edu/francois/software/tricubic/

3. Installation

Please see the pathways.asu.edu website for instructions on obtaining the source code. http://pathways.asu.edu .

frodaN is available as a binary and as a full source-code distribution. Each distribution will be compressed by using the standard UNIX applications TAR and GZIP. The names of the distribution files will differ depending on which distribution you are using, but they will adopt the following pattern:

> Binary distribution: `frodaN-ver#.tar.gz`
> Source distribution: `frodaN-ver#-src.tar.gz`

Please follow the appropriate instructions below for installing the binary distribution or the source code distribution.

3.1. Installing a binary distribution

The binary distribution of frodaN contains a compiled version of the program and the supporting directories and software that are necessary to run the program. Please follow step 3 carefully to ensure proper installation of the software.

1. Copy the file `frodaN-ver#.tar.gz` to the directory where you would like to install the software.
2. Untar the distribution by typing command:
   ```
   tar –zxvf frodaN-ver#.tar.gz
   ```
   Set an environment variable named `FRODANHOME` that contains the top-level directory of the frodaN distribution (for example, `/home/user/prog/frodaN-ver#`). To set the environment variable in bash shell, type command:
   ```
   export FRODANHOME=/home/user/prog/frodaN-ver#
   ```

3.2. Installing a source code distribution

1. Copy the file `frodaN-ver#-src.tar.gz` to the directory where you would like to install the software.
2. Untar the distribution by typing command:
   ```
   tar –zxvf frodaN-ver#-src.tar.gz
   ```
3. Change directory into `frodaN-ver#-src`
4. The program is compiled by using a Makefile. The Makefile is written to use g++ as the compiler. NOTE: If you plan to use a compiler other than g++, you can try to edit the Makefile, but we have not tested whether the code compiles with other compilers. The relevant line looks like this.
   ```
   CPP = g++
   ```

Change to:

```
CPP = name of your c++ compiler.
```

5. Set an environment variable named `FRODANHOME` that contains the top-level directory of the frodaN distribution (for example, `/home/user/prog/frodaN-ver#`). To set the environment variable in bash shell, type command:

```
export FRODANHOME=/home/user/prog/frodaN-ver#
```

4. Geometric Simulation with frodaN

This section describes geometric model and constraints used in frodaN. We refer the user to the Farrell et al. publication for more information about the concept and methodology. Some of the contents of this chapter have been taken verbatim from the Farrell et al. paper.

frodaN can be run in two ways: (1) to generate a pathway between two given structures subject to geometric constraints; (2) to explore conformational space subject to geometric constraints, starting from a given structure. Note that this is not molecular dynamics: there are no forces, accelerations, velocities, time steps, or energies. frodaN is a geometric simulation approach that samples conformations that adhere to a set of geometric constraints. Conformations are either allowed (meet the constraints) or disallowed (violate the constraints).

frodaN needs to several pieces of data about the system, provided as input. These can be easily generated using the preprocessing script (section 3.3), with just a PDB file as input:

- PDB information about the structure--initial coordinates of the atoms, atom names, residue names, atomic element names
- a list of covalent bonds
- an assignment of each atom to a numeric “atom type” (used for looking up pair-specific non-overlap cutoff distances)
- a grouping of atoms into rigid units.

4.1. Rigid Units

The rigid units used in frodaN are small and only account for the rigid covalent bond distances, covalent three-body angles, and rigid (non-rotatable) double bonds and peptide bonds. The largest rigid unit in the standard amino acids would be the indole group of tryptophan, and the smallest rigid unit would be the three-atom C-OH groups in several of the amino acids (This is different from the usual way that the software programs FIRST and FRODA assign rigid units, in which hydrogen bonds and hydrophobic contacts factor into the rigidity analysis, removing degrees of freedom and often producing large rigid clusters).

The rigid units are the objects that move, not the atoms. The atoms are still modeled and tracked but they are rigidly embedded in the rigid units. Certain atoms are “shared” or “split” across multiple rigid units, allowing an atom to maintain rigid-body relationships with multiple groups of atoms (see Farrell et al, Fig. 1).

4.2. Geometric constraints

frodaN establishes geometric constraints to govern the allowed and disallowed conformations of the system of rigid units. Note that no constraints are needed to maintain the covalent bond geometry within a rigid unit, because the rigid units are themselves the fundamental mobile units of the system:

Shared atom constraints

Shared atom constraints make shared atoms coincide (for atoms shared by multiple rigid units). This is accomplished by constraining the distance between each pair of shared atoms to be zero.

Hydrogen bond constraints

The initial structure is automatically examined to search for hydrogen bonds. To be a hydrogen bond, the atoms must have an "energy score" better than some cutoff energy (typically -1.0 kcal/mol) according to an energy function. Each hydrogen atom is only allowed to form one hydrogen bond (if there are multiple candidates, the bond with the best energy is chosen). For each identified hydrogen bond, a maximum distance constraint is established between the hydrogen and acceptor atom. The maximum distance is set to the initial H-A distance found in the input structure (but is not set less than 2.0 Å). Optionally, minimum angle constraints can also be turned on, which impose a minimum 100° donor-hydrogen-acceptor angular constraint. For backbone hydrogen bonds the minimum N-H...O angle constraint is set to 140° (only if the hydrogen bond already satisfies this constraint in the input structure), and a minimum H…O=C angle constraint of 130° (only if the hydrogen bond already satisfies this constraint in the input structure).

Hydrophobic contact constraints

The initial structure is automatically examined to search for hydrophobic contacting atom pairs. Only carbons and sulfurs are considered, and only in the side chains of residues Leu, Ile, Trp, Val, Phe, Met, Ala, Tyr. For nucleic acids, all carbons are considered hydrophobic. For non-nucleic acids and non-amino acids, hydrophobic carbons and sulfurs are those that only have C and H neighbors. A hydrophobic contact is any pair of hydrophobic carbons/sulfurs within a distance 3.9 Å. The imposed constraint is a maximum distance constraint, that the distance between the pair shall not exceed the initial distance plus 0.5 Å. There is no limit to the number of hydrophobic contacts that an atom can make in proteins. However, for nucleic acids, only one hydrophobic contact constraint is allowed between any particular pair of bases (chosen as the pair with closest distance). This rule for nucleic acids follows the logic of Fulle et al. but does not treat the constraints as rigid. Instead the constraint is an inequality distance constraint.

Non-overlap constraints

Minimum distance constraints are imposed between non-bonded atom pairs (fourth neighbors and higher). The minimum distance is pair specific, and depends on the types of atoms involved. Each atom is assigned a numeric atom type. A lookup table tells what the minimum distance is for each possible pair of atom types. The atom types and cutoff distances can be user-tailored. The default types and cutoff distances have been developed by calibrating against Amber simulations in work that is not yet published (Farrell, D.W., Mamonova, T., Kurnikova, M., Thorpe, M.F. unpublished). The default set is in the supplied pairTypeCutoffs_Jun2010 file. See Chapter 7 for file format.

Ramachandran constraints

Additional minimum distance constraints are imposed between certain pairs of atoms in adjacent residues to block off the outlier regions of the Ramachandran plot. This follows the work of Ho et al. These constraints override the non-overlap constraints. The constraint distances are listed in Table 1 of Farrell et al. frodaN currently only sets these constraints for atoms in residues having one of the 20 standard 3-letter residue names, and the atom names must also be standard.

Side chain torsion constraints

Additional minimum distance constraints are imposed between certain 1-4 bonded atom pairs (third neighbors) to keep them out of eclipsed configurations. These are only applied to 1-4 bonded atoms when atoms 2 and 3 are sp3-hybridized and have four neighbors each. For such 1-4 pairs, the minimum distance is calculated as the distance at which the atoms make a dihedral angle of 55°. The constraint is only applied to side chain atoms in the 20 standard amino acids with standard 3-letter names.

RMSD constraint

In targeting only, a constraint on the RMSD (root mean square distance) to the target structure is established. The RMSD is gradually decreased towards zero to induce a transition from the initial state to the target state. The RMSD is calculated over all targeted atoms (optionally, the RMSD can be over only the non-hydrogen heavy atoms).

Input structures take precent over predefined constraints

If any pairs of atoms in the initial structure (or target structure) are closer than is allowed by one of the predefined minimum distance constraints (non-overlap, Ramachandran, and side chain torsion constraints), the constraint is overridden and the constraint is set to the distance in the input structure.

4.3. Sampling Procedure (non-targeted, random perturbations)

In non-targeting sampling, there is really only one main decision to be made: whether to use random perturbations, or “momentum run-on” perturbatons. The random perturbations are large translations and rotations of the rigid units. However, because of the randomness, the sampling may be limited by the slow rate of diffusion. In other words, if you have some mobile domain, it may take many many random steps of all the tiny rigid units inside the domain before the domain itself has moved significantly. The momentum run-on perturbations do involve a small random component, but they can produce large-amplitude motions much more quickly than using random perturbations. Momentum run on is described in the next section.

The conformational sampling begins with the atoms in their initial positions from the input initial structure. For random perturbations (not momentum run-on), each step of the sampling consists of the following actions:

1. Randomly perturb the rigid units of the system. Each rigid unit is randomly displaced and rotated, without regard for any constraints. The size of the perturbation is rather large, on the scale of 1 Å for translational displacement and 120° for rotational motion, so that rigid units can hop over disallowed dihedral angle regions.
2. Enforce constraints. Constraints are enforced by minimizing a constraint-violation pseudo-energy function (see Farrell et al.)
3. Because the perturbation is large, some rigid units may get stuck during the enforcement of constraints. A search is made for any severe constraint violations (worse than 0.15 Å for maximum distance hydrogen bond or hydrophobic distance constraints, or worse than 0.07 for minimum distance constraints, including non-overlap, Ramachandra, and side-chain torsion). All residues with a severe constraint violation are reverted back to their original positions (where they were before step 1).
4. Re-enforce constraints if anything moved during step 3.
5. Global fitting. Finish the step by globally rotating and translating the entire system to optimize the RMSD to the initial structure.

frodaN repeats this for N steps, where N is some number specified in the input options to frodaN. frodaN will not automatically stop if for some reason constraints begein to be violated severaly. But frodaN does report the status of the constraints at each step, so this can be monitored.

4.4. Sampling Procedure (non-targeted, momentum perturbations)

Momentum steps are so named because the motion tends to persist in the same direction over many steps. Note that momentum is not actually conserved, since we are not integrating equations of motion, and there are no time steps or velocities. Here, the net translational and rotational change of each rigid unit is recorded for each step and used as a perturbation in the next step. Throughout the momentum steps, the upper-bound RMSD constraint is kept active, ensuring that the RMSD does not go back further. A momentum step involves the following actions:

1. Store current configuration. The six degrees of freedom of each rigid unit are stored in a $6M$-dimensional vector $\mathbf{q}_1$, where M is the number of rigid units.
2. Perturb rigid units by the last $\Delta\boldsymbol{q}$. The rigid units are translated and rotated by adding $\Delta\mathbf{q}$ from the previous momentum step, or 0 if this is the first momentum step. The system is now at a new configuration $\mathbf{q}_2$.
3. Small Random Perturbation. Randomly perturb the rigid units (translationally and rotationally), but do so with a very small amplitude (Atoms move by only about 0.05 Å). The system is now at $\mathbf{q}_3$.
4. Enforce constraints. Both the RMSD constraint and the structural constraints are enforced, bringing the system to state $\mathbf{q}_4$.
5. Global fit to initial structure. Remove any global translations and rotations by globally fitting to the target structure, bringing the system to state $\mathbf{q}_5$.
6. Calculate net change. Determine the net change of the degrees of freedom in this momentum step, $\Delta\mathbf{q} = \mathbf{q}_5 - \mathbf{q}_1$, for use in the next step. Then move on to the next step.

4.5. Sampling Procedure (targeted)

The targeting begins with the atoms in their initial positions from the input initial structure. The RMSD of the initial structure relative to the target structure, calculated over all targeted atoms, is some number C_0. An RMSD step size δ is chosen, typically 0.1 Å or less. Each targeting step consists of the following actions:

1. Advance the RMSD constraint. Set the RMSD constraint to $RMSD < C_i$, where $C_i = C_{i-1} - \delta$, where subscript i denotes the step number.
2. Enforce constraints. The RMSD constraint and structural constraints are enforced simultaneously, causing the rigid units of the system must move and rotate, often taking atoms in curved paths. See Farrell et al. for how contraints are enforced.

3. Global fitting. Finish the step by globally rotating and translating the entire system to optimize the RMSD to the target.
4. If structure is acceptable, move on to next step. The criteria for judging whether the structure is acceptable are that the non-overlap constraints not be violated by more than 0.2 Å, and that the shared atoms between adjoining rigid units not be more than 0.2 Å apart. In the most basic form of targeting, the targeting steps are terminated here if the structure is not acceptable. This can happen when the targeting has run up against a particularly difficult obstacle that it cannot find a way to get around without violating structural constraints.

Random Motion

The basic targeting procedure described above contains no random motion. The resulting pathway is deterministic, and atoms appear to move smoothly. To produce a random pathway, random motion can be optionally added to each targeting step as follows. At the beginning of each step, each rigid unit is randomly displaced and rotated, without regard for any constraints. The rest of the targeting step continues as usual. The constraint violations created by the random perturbation are restored during the “Enforce Constraints” portion of each step. The size of the perturbation is rather large, on the scale of 1 Å for translational displacement and 120° for rotational motion, so that rigid units can hop over disallowed dihedral angle regions. This can cause some rigid units to get stuck during the enforcement of constraints, in which case the problem rigid units are restored to their original positions and orientations.

Options for handling of hydrogen bond and hydrophobic contact constraints

“Common” hydrogen bond and hydrophobic constraints are those that are found in the initial and target structures. In the basic targeting procedure, the common constraints are kept fixed throughout the targeting under the assumption that the interactions are present during the entire pathway. As an option, the common constraints can be made breakable, or can be not included, instead of kept fixed. When a breakable constraint becomes stretched beyond a certain amount, it “breaks” and is removed. This can be helpful if some hydrogen bond or hydrophobic contact that is found in both structures needs to transiently break during the pathway. “Non-common” constraints are those that are in the initial structure but not in the final structure. The basic setting is to simply not include the non-common constraints since they are incompatible with the final structure. Optionally, the user can choose to include the non-common constraints as breakable constraints. Having the non-common constraints included may improve the quality of the pathways, since they preserve favorable interactions until the moment they break.

Recovery Methods

In the basic targeting procedure, if the shared-atom constraints and non-overlap constraints cannot be satisfied to within tolerance, the structure is deemed unacceptable and the targeting is terminated. Usually this does not happen until the very end, when the RMSD to target is quite low (<0.5 Å), and all the atoms are very close to their targets. It can sometimes happen earlier, when a particularly difficult obstacle in the pathway can cause the targeting to fail to produce an acceptable structure. A few recovery methods are available to try to help the protein move around the obstacle. The first is called “random retry”, which is to retry the last step using a random perturbation of the rigid units as described above in hopes that the random motion will help move past the obstacle. Typically up to 5 consecutive random retries are attempted.

Backtracking

Another available recovery method is “Backtracking.” In backtracking, the targeting steps switch into reverse , taking the RMSD away from the target instead of closer to the target. The sign of the RMSD step δ is reversed so that the RMSD constraint C_i increases instead of decreases at each step. The inequality in the RMSD constraint is also switched to a greater-than sign, $RMSD > C_i$, to carry the system away from the target. The idea is to go back in RMSD, find a new starting point at the higher RMSD level, then return to forward steps, in hopes that this enables the system to get around an obstacle. The method used to find a new starting point at the elevated RMSD before returning to forward steps is called “momentum run-on”, described in section 4.4. A preset number of momentum run-on steps are performed, with an upper-bound RMSD constraint so that the system does not wander away from the target in RMSD. The first time that a targeting step fails to produce an acceptable structure, the system is backtracked by 1 Å, a new starting point is found, and then the procedure returns to regular forward steps. If the targeting again gets stuck, the backtracking method tries going back by 2 Å, then 4 Å, then 8 Å, etc., doubling the amount of backward motion each time. The backtracking can even take the protein back in RMSD farther than the initial state. All non-common constraints are removed during backtracking so they do not hinder the system from going back in RMSD.

You should not normally run with backtracking enabled. Most of the time when the targeting stops, its because the system really is as close as possible to the target. However, in cases where you suspect that an obstacle is keeping the system from reaching the target, you can enable backtracking to try to circumnavigate the obstacle.

5. Preprocessing

Before running a simulation, several files must be generated that will be provided as input to frodaN. These files provide frodaN information about the system being simulated, such as the initial positions of atoms, atom names and elements, which atoms are covalently bonded to each other, and information about the geometric constraints that will be in effect during the simulation. A preprocessing script is included with the distribution to automate much of this work.

In this chapter, we describe the steps to follow before running frodaN: prepare PDB files, prepare a frodaN options file, run the preprocessing script to automatically generate data files needed by frodaN, or alternatively, manually generate the required data files if needed.

5.1. Prepare PDB files

You begin with a PDB file of the system that you want to simulate. If you are running a targeted simulations, you need two PDB files (initial structure and target structure). PDB files must be compliant with PDB format 3.2. Old PDB files (prior to October 2008) are not compliant—obtain new ones from the PDB. (see http://www.wwpdb.org/documentation/format32/v3.2.html.) Also, PDB files from many popular programs including Amber and CHARMM are not compliant, as they have different atom and residue naming conventions, and are often missing the element info that is part of PDB format 3.2, or are missing the TER separator between chains. The PDB atom numbers are not important in frodaN—they are ingored, so they can be anything (blanks, zeros, numbers, alphabetic characters).

First, check your PDB file to make sure it contains the atoms you want to simulate. For instance, often a PDB file contains multiple copies of the same chain, when you only want to simulate one of these chains. If the PDB file contains an NMR ensemble with multiple models, you should remove the extra NMR models and only include one model structure. We do not model missing atoms or missing residues for you, so you must do this yourself if you want these atoms in your system. Another problem that could arise is that the PDB file only contains atoms for one chain, even though the biological system should contain multiple chains, because the PDB file does not waste space specifying symmetry related atoms—in this case, if you want to simulate the whole biological system, you must create a PDB that has all the atoms explicitly listed (the PDB website can help you with this, but they provide symmetry-related atoms as if they were separate NMR models—you will need to change these so they are all in the same model, and give them different chain labels so that there is no redundancy of naming).

You must also decide whether to include heterogen atoms (non-standard amino/nucleic acids). In typical use, you will not want to simulate the waters, ligands, ions, etc. (geometric simulation is not molecular dynamics—there are no electrostatic forces, no solvation effects). You can remove heterogen atoms from your PDB file by hand, or let the preprocessing script do it for you (described later). If you do want to simulate heterogen atoms, be aware of the following: Waters, ions, or other single-atom groups are not supported, and the method currently has trouble handling these. You can keep ligands or non-standard amino/nucleic acids in your PDB file, but you should remove waters and single-atom groups (like counter-ions).

NOTE: frodaN gets its covalent bond information from FIRST. FIRST will only output the correct covalent bonding information for heterogens if the residue names and atom names match the PDB Het Dictionary.

Next, you must add hydrogens to your structure (not optional). The hydrogen atom names should also be compliant with the naming conventions of PDB format 3.2. We recommend using MolProbity to add hydrogens, available at http://molprobity.biochem.duke.edu/ .

If you are running a targeted simulation, two PDB structures are required (an initial state, and a final state). The target PDB file does not need to be a perfect match with the initial state—there can be residue insertions/deletions/mutations, or missing atoms. You should add hydrogens to the target structure just as you do with the initial structure.

5.2. Prepare frodaN options file

Next, you need to prepare a frodaN options file, or select one of the generic options files found in `$FRODANHOME/demo`, such as `options_fixedcons.xml` for a regular (non-targeted) run with fixed geometric constraints, or `options_target.xml` for a targeted run. See chapter 6 to tailor your own options file.

5.3. Run preprocessing script

The preprocessing script helps to create all required files to run the frodaN. It will also try to find and correct possible errors in the input files. Although it is hard to correct all possible errors, which can be, for example, caused by violation of PDB v3.2 format, we tried to identify and correct the most common ones.

You have to have Python 2.4 or newer installed on your machine to run the preprocessing script. The script is called preprocess.py, and is located in root directory of the software package.

For a regular (non-targeted) run, if your input structure with hydrogens is named input_H.pdb, and your options file is options.xml, you would type

```
$FRODANHOME/preprocess.py –i input_H.pdb –o options.xml
```

For a targeted run, with target structure target_H.pdb, type

```
$FRODANHOME/preprocess.py –i input_H.pdb –t target_H.pdb –o
options.xml
```

Be sure to add the `–nohet` command-line option if you want all heterogen atoms to be automatically stripped from your PDB files. (See note about this in section 5.1—you must at least remove waters and single-atom groups such as counter-ions).

Even though the options.xml file contains the frodaN simulation options, it is needed at the preprocessing stage because the preprocessing script uses it to see what data files it must create for input into frodaN.

preprocessing.py comand-line options:

Option	
-i	set initial PDB structure
-t	set target PDB structure. This option is required only for targeting run
-o	set run options XML file
-nohet	Exclude heterogens (HETATM records in PDB). By default heterogens will be included in calculations.
-help	

Output:
For the initial PDB structure specified by option –i, the following data files are generated. The files are named according to the filenames listed in the “modelfiles” section of the options.xml file.

- A processed PDB file, renumbered according to Hybrid-36 numbering scheme. Waters, ions, ligands, and HETATMs are removed, if the –nohet option is given.
- A covalent-bond topology file, according to FIRST format. See Chapter 7 for file format. Note that in FIRST format, the numbers in the file are atom indices ranging from 1..N, and do not necessarily correspond to the PDB atom ids.
- A rigid units file, according to FIRST format (see Chapter 7 for file format). This file specifies how atoms are grouped into rigid units.
- An atomtypes file. This file assigns a numeric atom type to each atom, for use in pair-specific repulsive interactions. (See Chapter 7 for file format)

For the target PDB structure (targeted runs only) specified by option –t, a set of data files is generated as above (processed PDB file, covalent bond topology file, etc.), with files named according to the filenames listed in the “targetfiles” section of the options.xml file. One additional file is generated:

- A target map file, that specifies which atoms from the initial structure are targeted to which atoms of the target structure. See Chapter 7 for file format. Special note: the atom indices in this file range from 0..N-1 for the initial structure with N atoms, and 0..M-1 for the target structure with M atoms. This target map is automatically created with the help of a sequence alignment.

5.4. Things that can go wrong during preprocessing

The input PDB files are run through FIRST to determine covalent bonding, which is used to determine rigid units. FIRST uses a PDB Het Dictionary to determine the bonding of residues with non-standard names. If your system has a residue that is non-standard, FIRST will not know how to bond the atoms, and it is likely that the covalent bonds will be incorrect, and consequently the rigid units. The covalent bond neighbor table is further used in frodaN, primarily to distinguish bonded atom pairs from non-bonded pairs (fourth neighbors and higher). With an incorrect covalent neighbor table, frodaN may think there is a severe clash and try to separate atoms that actually should be bonded.

For targeting, the preprocessing script performs a sequence alignment to determine which atoms from the initial structure correspond to atoms in the target structure. Sometimes the sequence alignment can mistakenly think that two residues correspond to each other when they do not. The resulting “map” file used by frodaN will then contain erroneous target information. This can be especially problematic for ligands. The sequence alignment may incorrectly pair a ligand (an unknown X residue in the sequence alignment) with a protein residue.

5.5. Manual Generation of Input Files

The preprocessing script is only for convenience. If desired, you can generate the input data files required by frodaN on your own. If you do this, you need to generate all files that are listed in the “modelfiles” section of your options.xml file. If you are doing a targeted run, you also need to generate all files that are listed in the “targetfiles” section of your options.xml file. See Chapter 7 for file formats.

6. Running frodaN

frodaN is executed in the following way:

```
$FRODANHOME/bin/main options.xml
```

where `options.xml` file contains options for the run in XML format. The following section contains the description of XML format used to control the frodaN job.

frodaN outputs status updates to the standard output device (usually the screen) at every step. Typically you will want redirect this output to a file to keep a record of the simulation. To redirect output, type

```
$FRODANHOME/bin/main options.xml >& myoutput.txt
```

where the `>&` indicate that standard output and standard error output are to be redirected to the listed file.

Also, usually you want to run frodaN in the background, and you want it to keep running even if you log off. To do this, type

```
nohup $FRODANHOME/bin/main options.xml >& myoutput.txt &
```

where `nohup` is the command for running something without stopping when the user logs off. The final `&` indicates that the run should be in the background.

6.1. XML format

Extensible Markup Language (XML) is a simple, very flexible format for organizing data within a text file. We chose XML for the frodaN options file rather than creating our own format, because it is a standard and widely-used format that allows for hierarchical organization.

For users unfamiliar with XML, we show below a sample, basic XML file to demonstrate its format.

```
<?xml version="1.0" encoding="UTF-8" ?>
<food>

        <Pizza crust="thin" size_inches="10" >
        </Pizza>

        <!-- this is a comment -->

        <Cookie
                type="chocolate chip"
                quantity="24"
                bake_temperature="350"
        />

</food>
```

The first line “`<?xml version` …” is required by all XML files.

The XML file contains a hierarchy of “elements”. The first element is “food”. It begins with a start tag <food> and ends with an end tag </food>.

Within food there is a Pizza element. The start tag for Pizza has more than just the word Pizza, in this case. It has two attributes: crust, and size_inches. The value of crust is “thin”, and the value of size_inches is “10”. All values must be enclosed by double-quotes (“ “) after the equals sign (=), including numerical values. The attribute/value pairs have been written all on the same line, but this is not required. The end tag is simply </Pizza>.

Next within the food element there is a comment which begins with `<!--` and ends with `-->`.

After the comment, the next element within the food element is Cookie. It has three attributes, each with an associated value. Notice that here there is no end tag </Cookie>. Instead, the single Cookie tag is both a start and end tag, because it ends in />. It doesn’t matter whether you end with a separate end tag (as in </Pizza>), or combine the start and end tags together as with Cookie. Also the spacing and indentation do not matter and are purely for visual ease in seeing the hierarchy of elements.

There can only be one top-level element (food, in this case), but this top-level element can contain many elements (Pizza, and Cookie).

For more info on XML, see http://en.wikipedia.org/wiki/XML

6.2. Options XML file

The options xml file is read by both the preprocessing script and by frodaN. The preprocessing script looks at it to see what input files it needs to generate for frodaN. Then frodaN uses it to gather data for the run and to run the simulation.

The options file must contain only one top-level element. The name of the top-level element is not checked, and can be anything. In the example options files supplied in the distribution (in `demo` directory), the top-level element is called root (starting with <root> and ending with </root>).

The options are organized into various XML elements. Depending on the type of run you do, different option elements may be required. We will describe each element, but first, below is an example options XML file for a simple, non-targeted run, using fixed geometric constraints:

```
<?xml version="1.0" encoding="UTF-8" ?>
<root>
        <modelfiles
                pdb="1CFC_processed.pdb"
                cov="cov.txt"
                rigidunits="rc.txt"
```

```
               atomtypes="atomtypes.txt"
        />
        <constraints
               pairtypecutoffs="pairTypeCutoffs_Jun2010"
               paircutoffscalefactor="0.95"
               hbondenergycutoff="-1.0"
        />
        <runfixedconstraints
               nsteps="500"
               domomentumpert="true"
        />
        <output
               outevery="1"
        />
        <random
               seed="auto"
        />
</root>
```

And here is a sample options file for a targeted run, which contains more elements than in the previous case:

```
<?xml version="1.0" encoding="UTF-8" ?>
<root>

        <modelfiles
               pdb="1CFC_processed.pdb"
               cov="cov.txt"
               rigidunits="rc.txt"
               atomtypes="atomtypes.txt"
        />
        <targetfiles
               pdb="1CFD_targprocessed.pdb"
               cov="targcov.txt"
               rigidunits="targrc.txt"
               atomtypes="targatomtypes.txt"
               map="targmap.txt"
        />
        <constraints
               pairtypecutoffs="path/to/file/pairTypeCutoffs_Jun2010"
               paircutoffscalefactor="0.95"
               hbondenergycutoff="-1.0"
        />
        <runtarget
               targheavyatomsonly="true"
               dynamicconstraints="true"
               delta="0.01"
               targ="random"
               dobacktrack="false"
               commonconstrainthandling="fixed"
               noncommonconstrainthandling="remove"
        />
        <output
               outrmsdfromlast="0.1"
        />
        <random
               seed="auto"
        />
</root>
```

The options elements can appear in any order. Recognized options elements are:

- `modelfiles` (required for all runs), specifying various data files relating to the input structure

- `targetfiles` (for targeted runs only), specifying various data files relating to the target structure;
- `constraints` (required for all runs), sets parameters of geometric constraints
- `runtarget` (required for targeted runs), sets parameters for the targeted simulation
- `runfixedconstraints` (required for non-targeted, fixed constraints run), sets parameters for the (non-targeted) fixed constraints run.
- `output` (required for all runs), controls output settings
- `random` (optional), sets the random seed.

6.3. `modelfiles` options

The `modelfiles` element is required for all runs. It specifies data files relating to the input structure (in targeted runs, these files relate to the initial structure, not the target structure). frodaN will look for these files in the current directory, unless a relative or full path to the filename is given. An example is shown below:

```
<modelfiles
        pdb="processed.pdb"
        cov="cov.txt"
        rigidunits="rc.txt"
        atomtypes="atomtypes.txt"
/>
```

The preprocessing script (section 5.3) can automatically generate all these files for you, from just a PDB file (there are some things you must do to prepare your PDB file first, see section 5.1). The names above are just a suggestion. At the preprocessing stage, you can decide what filenames you want, put the desired names into your options.xml file, and the preprocessing script will create files with these names. When frodaN runs, it will look for these filenames as input files, so they must exist at the time that frodaN is run.

XML Attributes:

`atomtypes="`*`filename`*`"`

> Default: (no default value)
> File name for the atom types file, which lists a numerical atom type assignment for each atom, used in determining pair-specific hard-sphere cutoff distances. This file is automatically generated with the preprocessing script (Chapter 5).

`cov="`*`filename`*`"`

> Default: (no default value)

File name of initial structure list of covalent bonds. This file is automatically generated with the preprocessing script (Chapter 5).

`rigidunits="`*`filename`*`"`

Default: (no default value)
File name of initial structure list of rigid units. This file is automatically generated with the preprocessing script (Chapter 5).

`pdb="`*`filename`*`"`

Default: (no default value)
File name of the “processed” initial structure PDB file produced by the preprocessing script, which must have hydrogens added and must meet the specifications stated in Chapter 5 of this manual. The preprocessing script (Chapter 5) can automatically do some of the work to get the PDB file in order. For targeting, the preprocessing script also can swap symmetry-related atoms in side chains (like the two oxygen atoms in a carboxyl group, or the indole ring of phenylalanine that has two equivalent flip states), choosing the swap that best matches the target structure.

6.4. `targetfiles` options

The `targetfiles` element is required for targeting runs. It specifies data files relating to the target structure. frodaN will look for these files in the current directory, unless a relative or full path to the filename is given. An example is shown below:

```
<targetfiles
        pdb="targprocessed.pdb"
        cov="targcov.txt"
        rigidunits="targrc.txt"
        atomtypes="targatomtypes.txt"
        map=”targmap.txt”
/>
```

The preprocessing script (section 5.3) can automatically generate all these files for you, from the initial structure and target structure PDB files (there are some things you must do to prepare your PDB files first, see Chapter 5). The names above are just a suggestion. At the preprocessing stage, you can decide what filenames you want, put the desired names into your options.xml file, and the preprocessing script will create files with these names. When frodaN runs, it will look for these filenames as input files, so they must exist at the time that frodaN is run.

At a minimum, the target PDB file is required, and the “map” file is required. The cov, rigidunits, and atomtypes files are recommended, as they are used to determine common constraints between the initial and target structures, and to alter any minimum distance (repulsive) constraints that may conflict with the target structure.

It is acceptable for the target structure to have insertions/deletions/mutations relative to the initial structure. There can even be missing atoms, and even just C-alpha atoms. What is important is the “map” file, which indicates which atoms from the initial structure correspond to which atoms in the target structure.

XML Attributes:

`atomtypes="`*`filename`*`"`

> Default: (no default value)
> File name for the atom types file, which lists a numerical atom type assignment for each atom, used in determining pair-specific hard-sphere cutoff distances. This file is automatically generated with the preprocessing script (Chapter 5).

`cov="`*`filename`*`"`

> Default: (no default value)
> File name of initial structure list of covalent bonds. This file is automatically generated with the preprocessing script (Chapter 5).

`map="`*`filename`*`"`

> Default: (no default value)
> File name of the target map file. The target map file establishes the correspondence of atoms between the initial and targeted structures. This file is automatically generated with the preprocessing script (Chapter 5).

`pdb="`*`filename`*`"`

> Default: (no default value)
> File name of the “processed” target structure PDB file produced by the preprocessing script. It should have hydrogens added, although this is not required. (It is required if the user wants frodaN to identify common hydrogen bond constraints between the initial and target structures). The PDB file must meet the specifications stated in Chapter 5 of this manual. The preprocessing script (Chapter 5) can automatically do some of the work to get the PDB file in order.

`rigidunits="`*`filename`*`"`

> Default: (no default value)
> File name of initial structure list of rigid units. This file is automatically generated with the preprocessing script (Chapter 5).

6.5. `runtarget` options

The `runtarget` element is required for targeting runs. An example is shown below for a simple targeting run with no random motion, to get a smooth pathway:

```
<runtarget
        targheavyatomsonly="true"
        dynamicconstraints="false"
        delta="0.1"
        targ="direct"
        dobacktrack="false"
        commonconstrainthandling="fixed"
        noncommonconstrainthandling="remove"
        outputconstraintlists=”true”
/>
```

Here is an example for a targeting run with random motion, dynamic constraints, and a smaller RMSD step:

```
<runtarget
        targheavyatomsonly="true"
        dynamicconstraints="true"
        delta="0.01"
        targ="random"
        dobacktrack="false"
        commonconstrainthandling="fixed"
        noncommonconstrainthandling="breakable"
/>
```

XML Attributes:

`commonconstrainthandling="fixed|breakable|remove"`

Default: `fixed`
Common constraints are the hydrogen bonds and hydrophobic contacts that are found in both the initial and target structures. By default these constraints are `fixed` (unbreakable), but they can be made `breakable` (i.e., they break when stretched), or not included (`remove`).

`delta="`*`rmsd-stepsize-in-Angstroms`*`"`
Default: (no default value)
Range: Any positive real number (recommended 0.1 for direct runs without random motion, and 0.01 for runs with random motion)
This is the size of the RMSD step. RMSD here is the root-mean-square distance to the target, calculated over all targeted atoms.

`dobacktrack="true|false"`

Default: `false`
Turn on/off the backtracking. See description in section 4.5.

`dynamicconstraints="true|false"`

Default: `false`
New hydrogen bond constraints and hydrophobic contact constraints will be added at every step as atoms come into contact. These dynamically-added constraints will also break randomly or if stretched. It is recommended that you set common constraints as "fixed", non-common constraints as "breakable", and turn on random motion if using this setting.

`noncommonconstrainthandling="fixed|breakable|remove"`

Default: `remove`
Non-common constraints are the hydrogen bonds and hydrophobic contacts that are in the initial structure but not the target structure. By default these constraints are not included (`remove`) because they are incompatible with the target, but they can be made `breakable` (i.e., they are included but break when stretched), or `fixed` (unbreakable).

`outputconstraintlists="true|false"`

Default: `false`
If true, frodaN will output the list of hydrogen bond constraints and hydrophobic contact constraints that are enforced as of the start of the simulation (in the usual case, this is just the common constraints, which are found in both structures). It will output these constraints to `bbhb.txt` (backbone hydrogen bond constraints), `hb.txt` (hydrogen bond constraints), `ph.txt` (hydrophobic contact constraints). It also outputs constraints identified in the initial structure, writing files as above but with file prefix `initial_*`, as well as constraints identified in the target structure using the prefix `target_*`.

`targ="random|direct"`

Default: `direct`
Choose the targeting style. `direct` is the most rapid form of targeting, producing a smooth pathway with no random motion. Atoms still travel in complicated/curved paths because of the geometric constraints that are enforced. `random` is for generating random pathways, by introducing random motion into each step towards the target.

`targheavyatomsonly="true|false"`

Default: `false`
Choose whether the RMSD constraint pulls on all targeted atoms (false), or just the heavy (non-hydrogen) atoms (true).

6.6. `runfixedconstraints` options

Use element `runfixedconstraints` to run a regular (non-targeted) exploration of conformational space subject to a fixed set of geometric constraints. An example is shown below:

```
<runfixedconstraints
        nsteps="500"
        domomentumpert="true"
/>
```

XML Attributes:

`nsteps="`*`integer-number-of-steps`*`"`

> Default: (no default value)
> Sets the number of steps for the run. Each step is a perturbation of the system, followed by an enforcement of constraints.

`domomentumpert="true|false"`

> Default: `false`
> Turn on/off the momentum perturbation (see section 4.4). If off, random perturbations are used (section 4.3).

6.7. `constraints` options

The `constraints` element is required for all runs. An example is shown below:

```
<constraints
     pairtypecutoffs="pairTypeCutoffs_Jun2010"
     paircutoffscalefactor="0.95"
     hbondenergycutoff="-1.0"
     hbondangleconstraints="true"
/>
```

XML Attributes:

`hbondenergycutoff="`*`energy-cutoff-in-kcal/mol`*`"`

> Default: `-1.0`
> Range: any real number ≤ 0
> Set the energy cutoff for including hydrogen bonds in the list of constraints (section 4.2). The value is in units of kcal/mol.

`pairtypecutoffs="`*`filename`*`"`

> Default: (no default value)

This file is the pair-specific lookup table for minimum distance constraints (to prevent overlap of non-bonded atoms). For example `pairTypeCutoffs_Jun2010`. frodaN will look in the current working directory for the specified file, and if not found, it will look in the $FRODANHOME/dat directory (this is where the cutoff tables that come with the program are located). You can use your own cutoff tables and your own set of atom types if desired.

`paircutoffscalefactor="`*`scale-factor`*`"`

Default: `1.00`
Recommended value: `0.95`
This setting applies a scale factor to all non-bonded cutoff distances. It is recommended that when using the pair type cutoff file `pairTypeCutoffs_Jun2010`, a scaling of 0.95 be applied.

6.8. `output` options

The `output` element is required for all runs. You can opt to have conformations output at regular intervals, or when the system has moved by a threshhold amount relative to the last output structure. An example is shown below for regular output:

```
<output
     outevery="1"
/>
```

Here is an alternative for output only after the system has moved by 0.5 Å RMSD from the last output structure:

```
<output
     outrmsdfromlast="0.5"
/>
```

XML Attributes:

`outevery="`*`output-every-Nth-conformation`*`"`

Default: 0
frodaN will output every Nth conformer in PDB file format. The default value 0 disables the regular output of PDB files. The output files are named `*_snapshot_xxxxxxxx.pdb`, where the * is the input PDB prefix from the `modelfiles` options element, and x stands for the snapshot number.

`outputrmsdfiles="true|false"`

> Default: `false`
> Turn on/off the output of root mean square deviation files. If on, a file called `RunningRMSD.txt` will be output that records the instantaneous RMSD of each output conformation relative to the initial structure, with RMSD calculated over C-alpha atoms (or P atoms for nucleic acids). Also, the cumulative RMSD of each residue will be output to files named `*_resrmsd_xxxxxxxx.txt`, where the * is the input PDB prefix from the `modelfiles` options element, and x stands for the snapshot number. In these cumulative RMSD files, a number is reported for each residue. This number is calculated as the root of the mean of the square distance of this C-alpha (or P) relative to the initial state, with mean taken over all previously output conformations. It is strictly speaking not the RMSF (root mean square fluctuation) of the residue, since the distance of the C-alpha is measure relative to the position in the initial structure, not the mean position over the trajectory.

`outrmsdfromlast="`*`real-number-in-Angstroms`*`"`

> Default: 0
> frodaN will output a new conformation when the system has moved by the specified distance in RMSD from the last output structure. The default value of 0 disables this output.

6.9. `random` options

The `random` element is optional and is used to set the random seed. The “seed” is an integer that initializes the pseudo-random number generator. If you want to reproduce a run, you simply run it again with the same random seed. The seed is output by frodaN to standard output at the beginning of each run, in case you want to re-run with the same random seed.

The random number generator used in frodaN is the Mersenne Twister (MT 19937)
http://www.math.sci.hiroshima-u.ac.jp/~m-mat/MT/emt.html

An example of the `random` options element is:

```
<random
     seed="auto"
/>
```

This will automatically set the seed by reading a random number from the hardware random number generator /dev/random in Unix systems, which aims to be a “true” random number generator that uses noise from the environment to set bits to random values. (See http://en.wikipedia.org/wiki//dev/random)

To explicitly set the seed to some value, use a hexidecimal number with 8 hex values (32 bits). (The “0x” is optional, as whatever is entered will be interpreted as hex).

```
<random
     seed="0x14d0c0ab"
/>
```

XML Attributes:

`seed="`*`8-hex-values`*`" or "auto"`

Default: `auto`
See discussion above.

7. Input File Formats

This section describes the format of the various input files required by frodaN. In typical use, you let the preprocessing script prepare input files for you. These file formats will be useful if you want to tailor your own input files.

7.1. Protein Data Bank

frodaN will recognize Protein Data Bank files consistent with the "PDB File Formats Contents Guide V3.20" or newer, which can be found at the PDB website (see online resources below).

Recognized File Extensions:

.pdb, .PDB, .ent

Example File:

```
HEADER    CALCIUM-BINDING PROTEIN                 02-AUG-95   1CFC
TITLE     CALCIUM-FREE CALMODULIN
ATOM      1  N   ALA A   1      14.550  12.461 -10.584  1.00  1.43           N
ATOM      2  CA  ALA A   1      14.376  13.393  -9.434  1.00  1.35           C
ATOM      3  C   ALA A   1      13.482  12.739  -8.378  1.00  1.11           C
ATOM      4  O   ALA A   1      13.926  11.920  -7.598  1.00  1.30           O
ATOM      5  CB  ALA A   1      13.724  14.689  -9.919  1.00  2.33           C
ATOM      6  H1  ALA A   1      14.247  11.506 -10.303  1.00  2.44           H
ATOM      7  H2  ALA A   1      13.973  12.787 -11.385  1.00  1.69           H
ATOM      8  H3  ALA A   1      15.550  12.440 -10.865  1.00  1.86           H
ATOM      9  HA  ALA A   1      15.340  13.615  -9.002  1.00  2.90           H
ATOM     10  HB1 ALA A   1      12.884  14.454 -10.555  1.00  2.16           H
ATOM     11  HB2 ALA A   1      13.384  15.260  -9.069  1.00  3.52           H
ATOM     12  HB3 ALA A   1      14.446  15.268 -10.476  1.00  2.98           H
ATOM   2240  N   LYS A 148      20.484   8.512  -6.399  1.00  2.87           N
ATOM   2241  CA  LYS A 148      21.250   7.294  -5.997  1.00  4.48           C
ATOM   2242  C   LYS A 148      20.642   6.708  -4.722  1.00  6.59           C
ATOM   2243  O   LYS A 148      19.599   7.188  -4.312  1.00  7.00           O
ATOM   2244  CB  LYS A 148      22.704   7.677  -5.735  1.00  4.10           C
ATOM   2245  CG  LYS A 148      23.338   8.196  -7.029  1.00  2.15           C
ATOM   2246  CD  LYS A 148      24.847   8.343  -6.833  1.00  2.51           C
ATOM   2247  CE  LYS A 148      25.441   9.115  -8.015  1.00  1.65           C
ATOM   2248  NZ  LYS A 148      24.984  10.532  -7.960  1.00  1.66           N
ATOM   2249  OXT LYS A 148      21.233   5.790  -4.175  1.00  7.97           O
```

Notes:

1. The following online resources have more information about PDB files, and the PDB file format:
   http://www.rcsb.org/
   http://www.wwpdb.org/docs.html
2. The PDB file format v3.2 has some limitations due to restrictions on field widths. One common problem is that the atom number field is only 5 characters long, so this file format only allows proper numbering for structures up to 99,999 atoms. frodaN does not look at the PDB atom numbers, using instead its own internal numbering from 0..N-1. So, in principle, you can use files with completely empty atom numbers if

desired. The preprocessing script replaces the input PDB atom numbers according to the hybrid-36 counting system http://cci.lbl.gov/hybrid_36/ .

3. The PDB file format v3.2 only allows 66 unique characters for labeling chains. If your file consists of more than 66 you can use hybrid-36 counting system.

7.2. map file

This file is used by frodaN to set the correspondence between atomic numbers found in initial and target structures of biomolecule.

NOTE: the map file does not use PDB atom numbers. It uses atom indices 0..N-1, where N is the number of atoms in the initial PDB file, and 0..M-1 where M is the number of atoms in the target PDB file. 0 represents the first atom in the PDB file.

By default the map file is created in the preprocessing step using information from a sequence alignment.

Example File:

```
0 0
1 1
6 10
7 3
8 20
```

Field Descriptions:

Field	Description
1	Atom index of the atom in the initial structure. Atom numbers are zero-based, running from 0..N-1, where N is the number of atoms in the initial PDB file.
2	Atom number of the atom in the target structure. Atom numbers are zero-based, running from 0..M-1, where M is the number of atoms in the target PDB file.

Fields are separated by any number of whitespace (tabs or spaces)

Note that some atoms in one structure may have no corresponding atom in the target structure. If you don't want certain atoms to be targeted, leave them out of this list. It is recommended that you leave hydrogens in the list even if you don't want to target them, because you can use the `targheavyatomsonly` option in frodaN to turn off targeting of hydrogens. The hydrogens are useful even if not targeted, because they help determine common and non-common hydrogen bond constraints between the two structures.

7.3. atomtypes file

This file contains the numeric atom types assignment. In principle you can devise your own definitions of atom types. The preprocessing script will assign numeric types based on the default definition of atom types (defined in Appendix A). If you want to override these atom types, make your own atomtypes file. If using your own set of cutoff distances, you will also need to make your own pair type cutoffs file.

Example File:

```
0 5
1 1
2 12
3 8
4 0
5 0
6 0
```

Field Descriptions:

Field	Description
1	Atom index 0..N-1, where N is the number of atoms in the PDB file.
2	Numeric atom type 0..M-1, where M is the number of atom types.

Fields are separated by any number of whitespace (tabs or spaces)

7.4. rigidunits file

frodaN needs to be told how the atoms are grouped into rigid units. As described in Chapter 4, in the normal way that frodaN is run, these rigid units represent only the rigidity of the covalent bond geometry. Currently, frodaN only accepts rigid unit information from a text file produced by FIRST. The preprocessing script runs FIRST with special settings so that the rigid units only take into account the covalent bond geometry, and the output rigid units file from FIRST is then used as input to frodaN. This file contains more information than is needed by frodaN. frodaN only looks at the first column (FIRST atom number) and third column (FIRST rigid cluster number). In FIRST, the rigid units are written without the sharing of atoms, although the sharing can be deduced by extending rigid units by one covalent bond neighbor. frodaN reads in the rigid clusters from the FIRST rigid cluster file and automatically extends them across covalent bonds to determine the set of rigid units.

NOTE: FIRST uses a 1..N numbering scheme, which is different from frodaN's 0..N-1 numbering scheme. When frodaN reads FIRST format files, it assumes the files were written using the FIRST 1..N numbering scheme.

You can find more information about this file in FIRST User's Manual at:
http://flexweb.asu.edu/software/first/user_guides/

This file corresponds to *_data.txt output file from FIRST.

Example File:

```
#   Raw data from FIRST
#   FIRST num    orig num       rigid    stressed   coll mode
# -------------------------------------------------------------------
           1           1          32          0           0
           2           2         493          0           0
           3           3          33          0           0
           4           4          33          0           0
           5           5          34          0           0
           6           6          32          0           0
           7           7          32          0           0
           8           8          32          0           0
           9           9         493          0           0
          10          10          34          0           0
          11          11          34          0           0
          12          12          34          0           0
```

Field sneeded by frodaN:

Field	Description
1	The FIRST atom number. FIRST assigns these numbers as 1..N, where N is the number of atoms.
2	The original atom ID as found in the input PDB file. frodaN ignores this ID, but it does need to be there.
3	The label of the rigid cluster to which this atom belongs. Every atom is assigned to exactly one rigid cluster (However, it is implicitly understood that these rigid clusters are to be extended by one covalent bond neighbor, which results in multiple rigid clusters for some atoms).
All other columns	Ignored

frodaN ignores lines in the FIRST rigid unit file that begin with #, and frodaN reads fields separated by any number of whitespace (tabs or space) characters.

7.5. cov file

frodaN needs to know the covalent bond neighbor table. Currently, it gets this information from an output file created by FIRST. In FIRST, there are two possible formats for this file—in one format, the numbers correspond to PDB atom IDs. This format will not work with frodaN, since PDB IDs are ignored. The other format produced by FIRST uses the FIRST internal atoms numbers 1..N.

NOTE: FIRST uses a 1..N numbering scheme, which is different from frodaN's 0..N-1 numbering scheme. When frodaN reads FIRST format files, it assumes the files were written using the FIRST 1..N numbering scheme.

You can find more information in FIRST User's Manual at:
http://flexweb.asu.edu/software/first/user_guides/

Example File:

```
1          2          5
1          6          5
1          7          5
1          8          5
2          3          5
2          5          5
2          9          5
```

Field Descriptions:

Field	Description
1	Atom 1 of the covalent bond. The number is the FIRST atom number, 1..N where N is the number of atoms.
2	Atom 2 of the covalent bond. The number is the FIRST atom number, 1..N where N is the number of atoms.
3	Ignored by frodaN. Can be empty.

frodaN reads fields separated by any number of whitespace (tabs or space) characters.

7.6. pairtypecutoffs file

frodaN needs to know the pair-specific cutoff distances that are to be used as constraints to prevent non-bonded overlap. These distances are listed in a pairtypecutoffs file. frodaN comes with a default file called `pairTypeCutoffs_Jun2010`. You can prepare your own pairtypecutoffs file if desired. frodaN uses the numeric atom types listed in the atomtypes file to know which type each atom is, and it uses the pairtypecutoffs file to get the cutoff distances.

The pairtypecutoffs file lists a set of radii for each atom type. These radii are additive. Optionally, it also lists pair-specific cutoff distances that override the additive radii. This makes it possible to have a unique cutoff distance for each possible pair of atom types that is not a simple sum of two radii. In the supplied pairtypecutoffs file `pairTypeCutoffs_Jun2010`, there are are 19 numeric atom types with pair-specific cutoffs defined between all possible pairs. These 19 are for standard protein atoms. There are an additional 11 numeric atom types for generic (non-protein) atom types

The format is the following (maybe later it will change to XML):

`30` (first line: the total number of atom types)

(next section: radius of each atom type. Required)

`0 1.7` (Field 1: atom type number. Field 2: atom radius)
`1 1.7`
`2 1.7`
…
`29 1.5` (the last line of the radii)

(next section: pair-specific cutoff distances.)
(these are optional, and it does not need to be a complete list)
(these cutoff distances override the sum of two radii for the given pair)

`0 0 3.45` (Field 1: atom type 1. Field 2: atom type 2. Field 3: cutoff)
`0 1 3.40`
`0 2 3.15`
…
`18 18 2.05`

8. Output Files

8.1. frodaN status file

frodaN outputs status messages to standard output. Usually, you want to redirect these to a file.

The beginning of the status file contains information about the initialization of the system. When the simulation begins, you see status messages like this:

```
Cycle 51
  Forward Step
  Attempting to add constraints
  Removed 1 stretched constraints
  Removed 20 randomly-chosen constraints
  Saving coordinates for Revert
  Doing random perturbation
  Performed 72 minimization cycles
  Reverted 2 problem residues
  Performed 34 minimization cycles
 0.041274 0.063559 0.035100 0.486955 0.116377 0.059199 0.060044   3.268724   3.265843  -
2.532090   2.062583
  Max Deviation 1287 13.8194
  Wrote snapshot 52
```

It is not practical to explain everything in the status output. However, one thing to explain is the line that looks like this

```
 0.041274 0.063559 0.035100 0.486955 0.116377 0.059199 0.060044   3.268724   3.265843  -
2.532090   2.062583
```

(the line is too long to fit onto this page, so it spills over onto the next line)

This list of numbers is a “constraint status line” reporting the state of the constraints after minimization. If you look early in the status file right when frodaN begins the simulation, you will see lines something like this:
(these will vary depending on the type of simulation you are running)

```
Columns:
1 Worst Shared-Point Distance
2 Worst Overlap Distance
3 Worst H-bond distance violation
4 Worst H-bond angle violation (deg)
5 Worst hydrophobic distance violation
6 Worst Rama distance violation
7 Worst side-chain torsion distance violation
8 RMSD constraint
9 RMSD to target
10 projection along linear target coordinate
11 projection along all orthogonal coordinates
```

This is telling you how to interpret the constraint status line. We see that the first number in the line is the “worst shared-point distance”. This refers to the shared atom constraints (see chapter 4), which are supposed to make shared atoms coincide. The number tells you the worst distance between any pair of shared atoms that are supposed to coincide, in this case, about 0.04 Å. A “good” number for this is below 0.1 Å. Anything above 0.1 Å means that the system is under some high strain, preventing the shared atoms from fully being brought back together.

The second number, worst overlap distance, tells you how big is worst violation of a non-overlap constraint. Ideally, there is zero overlap. A good number is typically below about 0.1 Å, meaning that the atoms are 0.1 Å closer than the constraint distance allows, which is acceptable.

The other “worst…” numbers tell you the worst constraint violations of the other types of constraints, all of them measured in Å, except for one. The worst H-bond angle violation is measured in degrees (a few degrees of violation is ok).

In targeting, you also see the current value of the RMSD constraint, and the actual RMSD to the target structure, measured over all targeted atoms.

Also in targeting, the “projection along linear target coordinate” is the projection of the 3N-dimensional position of the system relative to the target structure, projected onto the difference vector between the initial and target structures. The “projection along all orthogonal coordinates” tells you the the magnitude of all other (3N-1) degrees of freedom. These two numbers are essentially decomposing the RMSD into two components, scaled such that the sum of their squares is the instantaneous RMSD-to-target.

8.2. PDB Files

frodaN outputs trajectory files in PDB format. These files can be recognized as having the file extension “_snapshot_########.pdb” attached to the base name of the input file.

8.3. PDBaligned.seq

This file contains sequence alignment data. A sequence alignment is a way of arranging the sequences of DNA, RNA, or protein to identify regions of similarity. Deletion is shown as a dash symbol. Identical residues are marked by vertical bar.

PDBaligned.seq is not required to run frodaN, but it is created by preprocessing script and used to prepare map file. User can check what sequence alignment is used to find the atom match between initial and target structures.

Example File:

```
0     .    :    .    :    .    :    .    :    .    :
  A-LRGT---FSGFDGRADAEVLRKAMKGLGTDEDSILNLLTARSNAQRQQ
     |||   || || ||||| ||||||| ||||  ||  || | |||||
  KYTRGTVTAFSPFDARADAEALRKAMKGMGTDEETILKILTSRNNAQRQE
```

```
50     .    :    .    :    .    :    .
   IAEEFKTLFGRDLVNDMKSELTGKFEKLIVALMKP
   ||  |||||||||| | ||||||||| | | || |
   IASAFKTLFGRDLVDDLKSELTGKFETLMVSLMRP
```

Field Descriptions:

Line	Description
1	Sequential residue number. Sequence is split in 50 letters segments
2	Residue sequence of the initial structure
3	Residue match indicator. Vertical bar shows matching residues
4	Residue sequence of the target structure

9. Examples

9.1. Targeting run of calmodulin

The example can be found in `demo/targeting` folder of the distribution package.

This example shows the conformational transition of calcium-binding protein calmodulin. The initial state is shown in red color in Figure 1. Green color shows target state. Initial all-atoms RMSD between two states is 5.47 Ǻ. Figure 2 shows the change the RMSD during the targeting run.

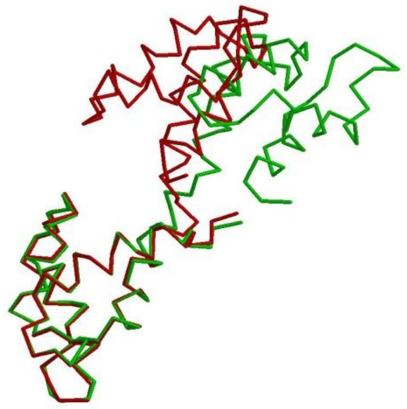

Figure 1. Overlap of two calmodulin structures.

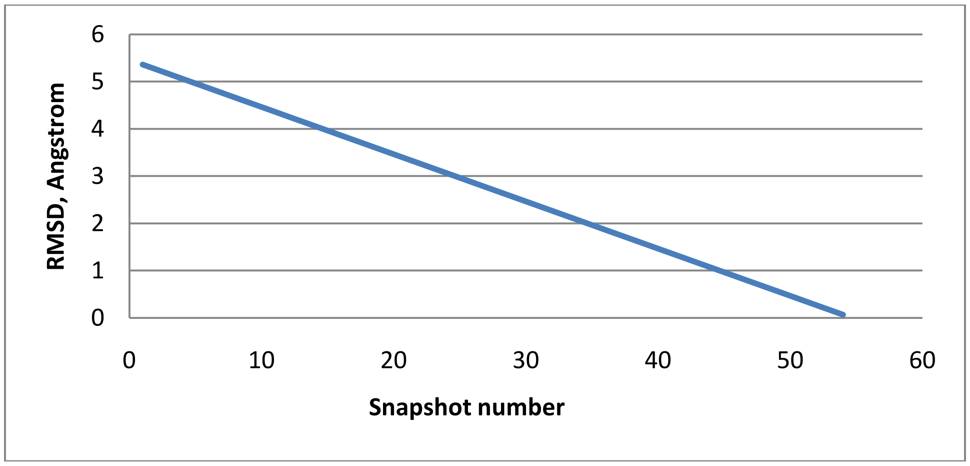

Figure 2. Change of RMSD during targeting run of calmodulin.

Procedure

1. Download the PDB file for initial calmodulin state (pdb code: 1CFC) from http://www.rcsb.org/pdb/explore/explore.do?structureId=1CFC.
2. Add protons to the structure by using the MolProbity online server (http://molprobity.biochem.duke.edu/). The structure file with protons was named 1CFC_FH.pdb.
3. Download the PDB file for target calmodulin state (pdb code: 1CFD) from http://www.rcsb.org/pdb/explore/explore.do?structureId=1CFD.
4. Add protons to the structure by using the MolProbity online server (http://molprobity.biochem.duke.edu/). The structure file with protons was named 1CFD_FH.pdb.
5. Create run options XML file. This file contains the options that will be used by frodaN. Below you can see an example of options file, you can find this example in demo/targeting/options.xml:

```
<?xml version="1.0" encoding="UTF-8" ?>
<root>
        <modelfiles
                pdb="1CFC_processed.pdb"
                cov="cov.txt"
                rigidunits="rc.txt"
                atomtypes="atomtypes.txt"
        />
        <targetfiles
                pdb="1CFD_targprocessed.pdb"
                cov="targcov.txt"
                rigidunits="targrc.txt"
                atomtypes="targatomtypes.txt"
                map="targmap.txt"
        />
        <constraints
                pairtypecutoffs="pairTypeCutoffs_Jun2010"
                paircutoffscalefactor="0.95"
                hbondenergycutoff="-1.0"
        />
        <runtarget
                targheavyatomsonly="true"
                dynamicconstraints="true"
                delta="0.01"
                targ="random"
                dobacktrack="false"
                commonconstrainthandling="fixed"
                noncommonconstrainthandling="remove"
        />
        <output
                outrmsdfromlast="0.1"
        />
        <random
                seed="auto"
        />
</root>
```

6. Set the FRODANHOME environment variable if not already set. In bash shell the command is
   ```
   export FRODANHOME=/home/user/prog/frodaN-ver#
   ```
7. Run preprocessing script. The command-line is:
   ```
   $FRODANHOME/preprocess.py -i 1CFC_FH.pdb -t 1CFD_FH.pdb -o options.xml
   ```
   Preprocessing script will create several files that are set in `options.xml` and required to run frodaN.
8. Run frodaN. The command-line to run in the background, with standard output redirected to a file is:
   ```
   nohup $FRODANHOME/bin/main options.xml >& frodan.out &
   ```

Results

Preprocessing script produces the number of files that are used in frodaN. The file names of newly created files are defined in `modelfiles` and `targetfiles` fields. Using these files frodaN produces output conformations in PDB format:

```
1CFC_processed_snapshot_000000001.pdb
1CFC_processed_snapshot_000000002.pdb
…
```

The RMSD between each successive pair of PDB conformations is at least 0.1Ǻ, which is controlled by `outrmsdfromlast` option in options.xml file.

9.2. Exploration of calmodulin conformational space

frodaN can also be used for exploration of the conformational space in biomolecules. This functionality is similar to FRODA run developed as part of FIRST software, see http://flexweb.asu.edu/software/first/user_guides/. In this regime frodaN will move the structural parts of biomolecule using the set of input constraints.

The example can be found in `demo/fixed_constraints` folder of the distribution package.

This example shows the sampling of the available conformational space of calcium-binding protein calmodulin. Figure 3 shows five overlapped structures obtained by running frodaN.

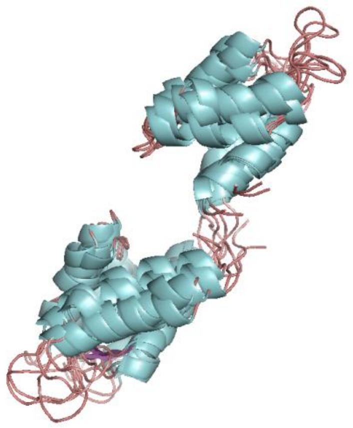

Figure 3. Five snapshots overlap of calmodulin structure.

Procedure

1. Download the PDB file for initial calmodulin state (pdb code: 1CFC) from http://www.rcsb.org/pdb/explore/explore.do?structureId=1CFC.
2. Add protons to the structure by using the MolProbity online server (http://molprobity.biochem.duke.edu/). The structure file with protons was named 1CFC_FH.pdb.
3. Create run options XML file. This file contains the options that will be used by frodaN. Below you can see an example of options file, you can find this example in demo/fixed_constraints/options_fixedcons.xml:

```
<?xml version="1.0" encoding="UTF-8" ?>
<root>
        <modelfiles
                pdb="1CFC_processed.pdb"
                cov="cov.txt"
                rigidunits="rc.txt"
                atomtypes="atomtypes.txt"
        />
        <constraints
                pairtypecutoffs="pairTypeCutoffs_Jun2010"
                paircutoffscalefactor="0.95"
                hbondenergycutoff="-1.0"
        />
        <runfixedconstraints
                nsteps="500"
                domomentumpert="true"
        />
        <random
                seed="auto"
        />
</root>
```

4. Set the FRODANHOME environment variable if not already set. In bash shell the command is
   ```
   export FRODANHOME=/home/user/prog/frodaN-ver#
   ```
5. Run preprocessing script. The command-line is:
   ```
   $FRODANHOME/preprocess.py -i 1CFC_FH.pdb -o options_fixedcons.xml
   ```
   Preprocessing script will create several files that are set in `options.xml` and required to run frodaN.
6. Run frodaN. The command-line is:
   ```
   $FRODANHOME/bin/main options.xml
   ```

Results

The preprocessing script produces the files that are used in frodaN. The file names of newly created files are defined in the `modelfiles` element of options.xml. Using these files frodaN produces 500 output conformations in PDB format:

```
1CFC_processed_snapshot_000000001.pdb
1CFC_processed_snapshot_000000002.pdb
…
1CFC_processed_snapshot_000000500.pdb
```

The number of conformations is controlled by `nsteps` option in options_fixedcons.xml file.

10. Appendix A

10.1. Definition of Atom Types

frodaN needs a numeric atom type for each atom and a lookup table with cutoff distances—in priniciple frodaN can accept any numeric atom types and any cutoff distances. Here we present how the numeric atom types are defined and set by the preprocessing script. We list the numeric value of the atom type first, followed by the description of the atom type.

PROTEIN ATOM TYPES
0 Amber CT type (sp3) with only C or H neighbors
1 Amber CT type (sp3) with at least one neighbor that is not C or H
2 Amber C type (carbonyl sp2 carbon)
3 Amber CA type (aromatic sp2 carbon in 6-membered rings and CE of Arg)
4 All other Amber Carbon types
5 Amber N type (sp2 Nitrogen in amides)
6 Amber N3 type (sp3 Nitrogen)
7 All other Amber Nitrogen types
8 Amber O type (sp2 oxygen in amides)
9 Amber O2 type (sp2 oxygen in anionic acids, COO-)
10 Amber OH type (sp3 oxygen with bonded hydrogen)
11 All other Amber oxygen types
12 All Amber S types
13 Amber H type (H attached to Nitrogen)
14 Amber HS type (H attached to Sulfur)
15 Amber HO and HW type (H attached to oxygen/water)
16 Amber HA type (H attached to aromatic carbon)
17 Amber HC type (H attached to aliphatic carbon with no electron-withdrawing substituents)
18 All other Amber H types

NON-PROTEIN ATOM TYPES
19 C
20 H
21 N
22 O
23 S
24 P
25 MG
26 MN
27 SI
28 FE

UNKNOWN ATOM TYPE
29 XX

11. References

Farrell, D. W., Speranskiy, K., & Thorpe, M.F.
Generating stereochemically acceptable protein pathways
Proteins: Structure, Function, and Bioinformatics (In Press, 2010)
doi: 10.1002/prot.22810
http://dx.doi.org/10.1002/prot.22810

Fulle, S., & Gohlke, H.
Analyzing the flexibility of RNA structures by constraint counting
Biophysical Journal (2008) **94**, 4202-4219.

Ho, B. K., Thomas, A., & Brasseur, R.
Revisiting the Ramachandran plot: Hard-sphere repulsion, electrostatics, and H-bonding in the alpha-helix
Protein Science (2003) 12:2508-2522.

Jacobs, D. J., Rader, A. J., Kuhn, L. A. & Thorpe, M. F.
Protein flexibility predictions using graph theory
Proteins-Structure Function and Genetics (2001) **44**, 150-165.

Wells, S., Menor, S., Hespenheide, B. & Thorpe, M. F.
Constrained geometric simulation of diffusive motion in proteins.
Physical Biology (2005) **2**, S127-S136.