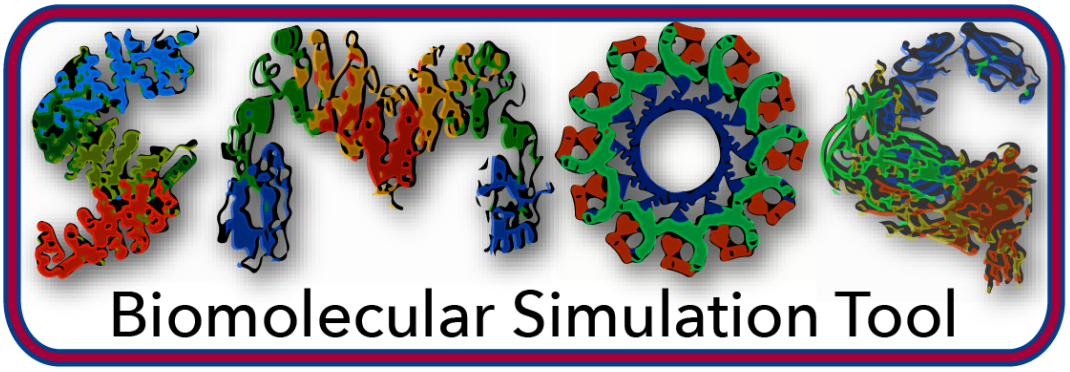
SMOG

Version 2.0
User's Manual

This manual will continually be updated

Last update: July 20, 2015

Rice University • Northeastern University



Authors:

Jeffrey Noel, Mariana Levi, Mohit Raghunathan

Heiko Lammert, Ryan Hayes, José Onuchic, Paul Whitford

info@smog-server.org

Should you read this manual?

If you would only like to use the basic functionality of SMOG v2 (i.e. the standard supported models), then you may find that the `README` file associated with the distribution provides all the information you need. This manual provides a more detailed description of the basic usage guidelines, in addition to advanced usage information and detailed descriptions of the underlying methodologies/models. For basic users, if the `README` is not sufficient, then Chapters 1, 2 and 3 will help you get started. For more advanced users, who may wish to modify structure-based models (e.g. extending to new residue types, ligands, electrostatics, etc), then consulting Chapters 4 and 5 will be necessary. We additionally provide appendices that have technical details that may be of interest to some users. While we try to provide all pertinent information here, don't hesitate to contact us for clarification.

Contents

Chapter 1

Introduction

1.1 What are Structure-Based Models?

Structure-based models (i.e. SBMs, or SMOG models) define a particular known conformation as a potential energy minimum. With this being the only requirement, there is an endless number of ways in which one may construct a structure-based model. For example, one may build protein-specific and RNA-specific variants, the resolution of the model can be varied, multiple minima may be included, and the degree to which non-native interactions are stabilizing can be adjusted. The utility of these models is equally diverse, where they may be applied for understanding dynamics, or for structural modeling objectives, as discussed in recent review articles [1, 2][1]. With such flexibility, variations within this general class of models can be tailored to ask specific questions about biomolecular processes. In the present document, we describe a set of computational tools that allows one to use previously-developed structure-based models, as well as design and implement new variations that are suited for your specific needs.

In the simplest form, a structure-based model defines a single configuration as the global potential energy minimum, where all intra- and inter-molecular interactions as assigned minima that correspond to that structure. This fully native-centric variant of the model is colloquially referred to as a "vanilla" structure-based model. In terms of the energy landscapes of biomolecules, these vanilla models represent an energetically unfrustrated landscape [3, 4]. Since biomolecular landscapes possess some degree of energetic roughness, it is often desirable to extend structure-based models to include both native and non-native interactions. As such, in the SMOG v2 software package, we provide two energetically unfrustrated models by default, upon which additional interactions may be added by the user. Specifically, in this software package we provide the coarse-grained

[1] [1] available at http://smog-server.org/noel/book_chapter_sbm.pdf

C_α structure-based model for proteins, as developed by Clementi et al [5]. We also provide the all-atom structure-based model, as developed by Whitford et al. [6]. While the C_α model is only defined for proteins, the all-atom model supports proteins, RNA, DNA and some ligands. While a complete description of the C_α model is available in the original reference, there have been a number of extensions in the all-atom model over the last several years. Accordingly, a complete description of the energetic parameters are given in Appendix A.

1.2 What does SMOG v2 do?

SMOG v2 is a software package designed to allow the user to start with a structure of a biomolecule (i.e. a PDB file) and construct a structure-based model, which is then simulated using Gromacs [7], or NAMD [8]. We previously implemented an online server (http://smog-server.org, [9]) that was capable of providing the vanilla flavor of structure-based models, along with a few adjustable parameters (i.e. SMOG v1). SMOG v2 is a complete rewrite of the original software package, and it provides four major advantages over its predecessor:

- Extensibility – One may add new residue and molecule types without source-code modifications.
- Portability – By building forcefield definitions on generally-defined XML-formatted files, researchers may easily distribute and share new SMOG model variants.
- Generalizability – Every energetic parameter may be varied, and additional energetic interactions (even non-native) may be included.
- Multi-resolution capabilities – All structural resolutions may be implemented, as well as multi-resolution variations.

It is important to note that none of these new features require additional programming, or source-code extensions. Rather, one simply adjusts the XML template files when designing SMOG variants. Further, the templates are not statically-linked to SMOG v2, allowing the user to easily choose from a library of models at runtime.

Chapter 2

"Installation"

Since SMOG v2 is written in Perl and Java there is no need for compilation and installation. However, one must configure a few settings and ensure that appropriate modules are available at runtime. This section describes the steps necessary to configure and verify that SMOG v2 is functioning properly on your local machine.

2.1 Prerequisites

SMOG v2 runs on all Unix-like operating systems. The prerequisites for SMOG v2 are
`Perl Programming Language`
`Perl Data Language (PDL)`,
as well as the following modules, which are available through the Perl module managing utility `CPAN`(recommended), or through manual installation:
`String::Util`
`XML::Simple`
`Exporter`
`XML::Validator::Schema`

Finally, your machine must have `Java Runtime Environment v1.7` or greater.

2.2 Configuration

Before running SMOG v2, you must configure it on your local machine. This is accomplished through a short two-step process:

1) Set the required environmental variables. To do so, modify the file `configure.smog2`, which is included with the distribution. Specifically, you will need to modify the following two lines:
`smog2dir=""`
`perl4smog=""`
`smog2dir` is the main SMOG directory, and `perl4smog` is the version of perl that you would like to use. On most linux systems, the default location of Perl is `"/usr/bin/perl"`, whereas on OSX it is typically `"/opt/local/bin/perl"`

2) Initialize the new environment variables with:

```
> source configure.smog2
```

This will set the required environment variables for your current session.
To automatically configure SMOG at login, you may want to add the above command to your shell profile file (e.g. `~/.bashrc`):

```
source /full-smog2-path/configure.smog2
```

and add

`/full-smog2-path/configure.smog2/bin` to your `PATH`.

3) Verify that `java` is installed (example output below)

```
> java -version
java version 1.7.0_25
Java(TM) SE Runtime Environment (build 1.7.0_25-b15)
Java HotSpot(TM) 64-Bit Server VM (build 23.25-b01, mixed mode)
```

If java is not found, make sure to install the JRE or JDK 1.7 or greater and that it is in your path (accessible as: `> java`).

2.3 Verify SMOG is properly configured

If SMOG is properly configured, then you will be able to run smog with the following command:

```
> smog2
```

If configuration was successful, then you will be greeted with the following message:

```
*****************************************************************************************
******* ******* ******* ******* ******* SMOG v2.0 ******* ******* ******* ******* *******
              Thank you for using the Structure-based Model (SMOG) software

    This package is the product of contributions from a number of people, including:
                      Jeffrey Noel, Mariana Levi, Mohit Raghunathan,
                         Ryan Hayes, Jose Onuchic & Paul Whitford

                    Copyright (c) 2015, The SMOG development team at
                       Rice University and Northeastern University

                SMOG v2.0 & Shadow are available at http://smog-server.org

                          Direct questions to: info@smog-server.org

*****************************************************************************************
```

In addition to verifying that SMOG v2 will start, it is highly recommended that you also run the testing scripts provided as a tarball (`smog-check.tar`), which is available at smog-server.org.

`smog-check.tar` contains two main test scripts. One script is very fast, whereas the second is very comprehensive and can be used to test new SMOG models that you may design. When everything works well, performing the the checks is as easy as issuing two commands. Just make sure you run config.bash before running the tests.

While in the directory `smog-check`, issue the command:

```
./quick-check
```

For the comprehensive check (may take up to 30 minutes to complete):

```
./smog-check
```

If you find that either script reports failures, please communicate that to the smog-server.org team, so that we may help diagnose the problem.

Chapter 3

Using SMOG v2

This chapter describes the usage of SMOG v2. It is recommended that all users read this chapter before using the software.

3.1 Preparing the input PDB file

3.1.1 PDB file format

To prepare a SMOG model, a structural model (e.g. crystallographic, NMR, or cryo-EM model) must be provided as a PDB file, in accordance with the PDB Content Guide, page 187.

To avoid I/O issues, please follow these additional guidelines when preparing your PDB file for use with SMOG v2:

- Only use a text editor (e.g. vi, or emacs) to prevent insertion of hidden characters.
- Only include lines that start with ATOM, HETATM, COMMENT(may be at the beginning, or end of any chain), BOND (user-defined specific bonds. Must appear after END), TER (to indicate a break between 2 chains) and END. Only BOND and COMMENT lines may appear after END.
- Chain identifiers are ignored. If your system has multiple chains, insert TER lines (left justified) between chains. NOTE: Do not immediately follow a TER line with an END line. This is interpreted as a chain with 0 atoms, and an error message will be issued.

- Only residues, and atoms within a residue defined in the forcefield templates will be recognized by SMOG v2. Unless a coarse-grained template is designated with -tCG, unrecognized residues and atoms will lead to a PDB parse error, and the program will exit.

3.1.2 Preprocessing

As discussed in Chapter 4, SMOG v2 reads "template" files in order to generate forcefield files. As such, each PDB file has to fully conform to the molecular structure definitions provided by the templates. For example, the default all-atom templates (provided in SBM_AA) distinguish between terminal and non-terminal residues (i.e. in proteins there is an OXT in place of a peptide bond for terminal residues). In the default templates, the terminal amino acid residues have an suffix "T" added to their their three-letter code (*e.g.* GLY vs. GLYT).

A preprocessing tool (`smog_adjustPDB`) is provided that will adjust your PDB to reflect changes necessary to conform to the templates. Assuming `$SMOG_PATH/bin` is in your `PATH`, the preprocessing tool may be run with the following command :

```
> smog2_adjustPDB <PDB file> <default | -f mapFile> <outputPDB.pdb>
```

The first required input is the PDB file, the second required input is either the option `default` to use the map file provided with the program (for use with the default SMOG models), or the option `-f` followed by your own map file. The map file should be formatted as follows:

```
#/mapFile
#<Residue> <head-terminal> <tail-terminal>
ALA ALA ALAT
G G5 G
...
...
```

Lines containing a "#" character are interpreted as comments. Each line must have three strings that are space/tab delimited. The first field is the original residue name, as it appears in the original PDB file. The second is the name to be substituted if the residue is the first residue in a chain, and the third field is the corresponding substitution for the last residue in each chain. The preprocessing tool will write a modified PDB file

outputPDB.pdb.

The script also numbers atom and residue indices to be sequential within each chain, and it adjusts atom names to be consistent with the `SBM_AA` template files.

3.2 Generating a Structure-Based Model

SMOG v2 supports a broad range of structure-based models, and the All-Atom [6], and the C_α [5] models are provided as defaults. See Appendix A for full details of the default models. By running SMOG, you will generate the .top, .gro, and .ndx files necessary to perform a structure-based simulation in Gromacs or NAMD. Additional output files are provided, for your information.

3.2.1 Default All-Atom Model

The all-atom potential energy function is defined through the template files found in the directory `$SMOG_PATH/SBM_AA`. These files define:
1) the covalent geometry of amino acids, nucleic acids, some ligands, as well as bioinorganic atoms.
2) the energetic and system parameters (e.g. mass, charge, interaction strengths).

To generate all-atom forcefield and coordinate files for the default model (i.e. .top and .gro files), issue the command:

```
> smog2 -i yourFile.pdb -AA
```

where *yourFile.pdb* is the name of the file containing your molecular system.

If you would like to specify a different all-atom model, then use the command:

```
> smog2 -i yourFile.pdb -t templateDirName
```

where `templateDirName` is the name of the directory containing the desired template files.

3.2.2 Default C_α model

To generate forcefield and coordinate files for the default C_α model, issue the command:

```
> smog2 -i yourFile.pdb -CA
```

If you would like to use a different set of CG templates:

```
> smog2 -i yourFile.pdb -t templateDirName -tCG CGtemplateDirName
```

Note that an additional set of templates are required when using a coarse-grained model. The option `-tCG` is used to indicate the precise coarse-grained model that should be prepared. When `-tCG` is given, then the `-t` flag is used to designate the templates that initially process the PDB for contact analysis. Normally an all-atom PDB is provided, since native contact maps make the most sense when generated from an all-atom structure (note that the "Shadow" map *only* makes sense with atomic graining). The `-tCG` templates are then used to construct the CG energetic model. In the above example, the PDB has residues and atoms corresponding to the `-t` templates, and these definition will also be used for contact map generation. See Appendix B for a detailed description of the supported contact map calculations.

3.3 Input options

SMOG v2 always requires a PDB file and some argument indicating that model should be used. Table 3.1 shows the currently-supported input arguments.

3.3.1 User-provided contact map

If you have generated contacts yourself, these can be used instead of using the internal SMOG2 routines. A single file containing all the contacts in a list can be specified at the command line with the switch `-c`. For example:

```
> smog2 -i <pdbfile> -c contacts.txt ...
```

will read the list of contacts in file `contacts.txt`

```
chainNum_i1 atomNum_i1 chainNum_j1 atomNum_j1 (opt.  distance)
chainNum_i2 atomNum_i2 chainNum_j2 atomNum_j2 (opt.  distance)
```

Input Option	**Usage**	**Default value**
Required		
`-i <file name>`	Input PDB file to define the model	none
Optional		
`-t <Folder Name>`	Folder containing templates of molecular and interaction definitions	none
`-AA`	Use the default all-atom model	N/A
`-CA`	Use the default C_α model	N/A
`-tCG <Folder Name>`	Folder containing templates used for coarse graining. Only necessary when CG enabled	none
`-c <string>`	Input contact file name	none
`-g <string>`	Output .gro file name	smog.gro
`-o <string>`	Output .top file name	smog.top
`-s <string>`	Output .contacts file name	smog.contacts
`-n <string>`	Output .ndx file name	smog.ndx
`-dname <string>`	Default name to use for all output files.	none
`-backup [yes/no]`	Enable/disable generation of backed up outputs.	none
`-warnonly`	Report fatal errors as warnings	N/A
`-limitbondlength`	If bond length exceeds limits, set it to the limiting value	N/A
`-limitcontactlength`	If contact length exceeds limits, set it to the limiting value	N/A

TABLE 3.1: Flags supported by SMOG v.2.0

```
chainNum_i3 atomNum_i3 chainNum_j3 atomNum_j3 (opt.  distance)
etc ...
```

which should be formatted as a single line per contact, whitespace delimited, where each line has the two atoms interacting and their respective chain numbers. The chains are numbered starting from 1 by the order of occurence in the PDB file. The atomNum should be consistent with atom numbers in the input PDB file. The fifth column can contain a numeric distance in Å which if provided will be used instead of the native distance. If using `-tCG` to obtain a coarse grained topology, the input contact map should designate residue numbers instead of atom numbers, again with the same numbering in the input PDB file.

3.4 Performing a simulation in Gromacs/NAMD

Once you have generate the .top and .gro files with SMOG, you are ready to perform a simulation. Rather than write a new molecular dynamics simulation package, SMOG generates input files for use with Gromacs[7] and NAMD [8], two highly-optimized and parallelized MD software suites. This allows you to use nearly every protocol that has been implemented in these programs when performing simulations with structure-based models (e.g. replica exchange, umbrella sampling). In addition, both of these packages are scalable to many processors through a combination of MPI and thread-based parallelization, allowing SMOG models to fully take advantage of cutting-edge computing resources. Here, we provide brief descriptions of how to perform SMOG model simulations in Gromacs and NAMD. More complete resources on performing these types of simulations may be found at http://smog-server.org in the NAMD manual.

3.4.1 Gromacs 4.5 or 4.6

3.4.1.1 All-Atom Model

First, produce a portable xdr file (in this case, run.tpr) that describes your simulation. This file is platform independent and contains all parameters for your simulations. This allows you to produce a tpr file on any machine, and then move it to another machine and run your simulation. The xdr file is produced by grompp (part of the Gromacs distribution):

```
> grompp -f mdpfile.mdp -c gro_file.gro -p top_file.top -o run.tpr
```

The file `mdpfile.mdp` tells Gromacs what settings to use during the simulation, such as the timestep size, the number of timesteps and what thermostat to use. Here is a sample set of configurations that are consistent with the default all-atom model:

```
integrator = sd ;Run control: Use Langevin Dynamics protocols.
dt = 0.002 ;time step in reduced units.
nsteps = 100000 ;number of integration steps
nstxout = 100000 ;frequency to write coordinates to output trajectory .trr file.
nstvout = 100000 ;frequency to write velocities to output trajectory .trr file
nstlog = 1000 ;frequency to write energies to log file
nstenergy = 1000 ;frequency to write energies to energy file
nstxtcout = 1000 ;frequency to write coordinates to .xtc trajectory
xtc_grps = system ;group(s) to write to .xtc trajectory (assuming no ndx file is supplied to grompp).
energygrps = system ;group(s) to write to energy file
nstlist = 20 ;Frequency to update the neighbor list
coulombtype = Cut-off
```

```
ns_type = grid ; use grid-based neighbor searching
rlist = 1.2 ;cut-off distance for the short-range neighbor list
rcoulomb = 1.2 ; cut-off distance for coulomb interactions
rvdw = 1.2 ; cut-off distance for Vdw interactions
pbc = no ; Periodic boundary conditions in all the directions
table-extension = 10 ; (nm) Should equals half of the box's longest diagonal.
tc-grps = system ;Temperature coupling
tau_t = 1.0 ; Temperature coupling time constant. Smaller values = stronger coupling.
ref_t = 120.0 ; In reduced units (see Gromacs manual for details)
Pcoupl = no ;Pressure coupling
gen_vel = yes ;Velocity generation
gen_temp = 50.0
gen_seed = -1
ld_seed = -1
comm_mode = angular ; center of mass velocity removal.
```

LISTING 3.1: Sample mdp file for all-atom SMOG models used for Gromacs v4.5/4.6

After you have generated the .tpr file with `grompp`, you will need to perform the simulation. To run the simulation, issue the command:

```
> mdrun -s run.tpr -noddcheck
```

Is is highly recommended that you explore all Gromacs options, in order to ensure maximum performance (e.g. the number of threads being used). **SMOG-model specific requirement:** To use domain decomposition when performing a simulation in parallel, using either threads, or MPI, you should add the additional flag `-noddcheck`. Note, that for protein folding you will probably want to avoid domain decomposition, and instead use particle decomposition by adding the option `-pd` when on a single node.

3.4.1.2 C_α Model

To run a simulation with the C_α model, the steps are the same as for the AA model, though there are a few minor changes. First, when running grompp, you will want to change a few settings in the .mdp file. A sample .mdp file for C_α models is given below.

```
integrator = sd ;Run control: Use Langevin Dynamics protocols.
dt = 0.0005 ;time step in reduced units.
nsteps = 100000 ;number of integration steps
nstxout = 100000 ;frequency to write coordinates to output trajectory .trr file.
nstvout = 100000 ;frequency to write velocities to output trajectory .trr file
nstlog = 1000 ;frequency to write energies to log file
nstenergy = 1000 ;frequency to write energies to energy file
nstxtcout = 1000 ;frequency to write coordinates to .xtc trajectory
xtc_grps = system ;group(s) to write to .xtc trajectory (assuming no ndx file is supplied to grompp).
energygrps = system ;group(s) to write to energy file
nstlist = 20 ;Frequency to update the neighbor list
coulombtype = Cut-off
ns_type = grid ; use grid-based neighbor searching
```

```
rlist = 3.0 ;cut-off distance for the short-range neighbor list
rcoulomb = 3.0 ; cut-off distance for coulomb interactions
rvdw = 3.0 ; cut-off distance for Vdw interactions
coulombtype = User
vdwtype = User
pbc = no ; Periodic boundary conditions in all the directions
table-extension = 10 ; (nm) Should equals half of the box's longest diagonal.
tc-grps = system ;Temperature coupling
tau_t = 1.0 ; Temperature coupling time constant. Smaller values = stronger coupling.
ref_t = 120.0 ; In reduced units (see Gromacs manual for details)
Pcoupl = no ;Pressure coupling
gen_vel = yes ;Velocity generation
gen_temp = 50.0
gen_seed = -1
ld_seed = -1
comm_mode = angular ; center of mass velocity removal.
```

LISTING 3.2: Sample mdp file for C_α SMOG models used for Gromacs v4.5

The most significant difference is the use of "User-defined" Vdw and Coulomb interactions. This is due to the fact that the 10-12 potential used for contact interactions in the C_α model. In order to run `mdrun` (next step), it is necessary to generate table files that define the 10-12 interaction. We provide a tools for generating these tables (`$SMOG_PATH/bin/smog_tablegen`) with the SMOG2 distribution. The table can be generates in a single step

```
$SMOG_PATH/bin/smog_tablegen <M> <N> <ion conc.> <elec.  switch dist.> <elec.
truncate dist.> <table length> <output name>
```

M the exponent on the attractive term, N is the exponent on the repulsive term. If you are not including electrostatics (most common), then provide values of 0 for `<ion conc.>`, `<elec.  switch dist.>` and `<elec.  truncate dist.>`. `<table length>` indicates how long the table should be, in nanometers. It is important that the table is longer than any native contacting pair of atoms may be during the simulation. Finally, the last argument is the filename for the table file. If you don't use a .xvg suffice, the script will add one for you.

After you have generated your tabulated potentials for the 10-12 interaction (i.e. table_file.xvg), and you have prepared a .tpr file with `grompp`, you can run the simulation with the command:

```
> mdrun -s run.tpr -noddcheck -table table_file.xvg -tablep table_file.xvg
```

Typically, for protein folding, you will want to avoid domain decomposition and instead use particle decomposition by adding the option `-pd` when on a single node. After you perform your simulation, you can utilize any analysis tools provided with Gromacs.

3.4.1.3 Examples

Check $SMOG_PATH/examples/gromacs4 for some complete examples with terminal history.

3.4.2 Notes and Hints

3.4.2.1 Domain Decomposition

The `[ pairs ]` section is treated as bonded by Gromacs and therefore all pairs within a single domain are always calculated. If you are using `-pd` with version 4.X or only OpenMP threads in version 5.0, you can reduce the cutoffs to values that ignore the pair distances and only take into account the non-bonded excluded volume (provided you have no electrostatics of course). For the default models this would be 0.6 for `-AA` and 1.0 for `-CA`.

3.4.3 Gromacs 5

Gromacs 5 has a few changes that impact SMOG models. First, we don't yet provide a Gromacs 5 distribution with the SMOG enhancements (umbrellas, `g_kuh`, gaussian contact potentials). So, if you want to use these you can only use Gromacs 4.5. Gromacs 5 itself has changes of note: 1) OpenMP support has replaced the option of particle decomposition and 2) OpenMP requires `cutoff-scheme=Verlet` and Verlet doesn't yet allow tabulated potentials. This has the largest impact on C-alpha models, which use tabulated potentials. If your simulated system has less than roughly 100 atoms, you can typically only use a single processor with v5, because additional threads are only allowed through OpenMP. If your system is large enough you can use multiple MPI processes with domain decomposition to scale to multiple cores. When using Verlet lists you have to use `pbc = xyz`. For all-atom simulations, Verlet lists are fine, and it is usually best to use as many OpenMP threads as possible with `-ntomp`.

3.4.3.1 Examples

Check $SMOG_PATH/examples/gromacs5 for complete examples, including terminal history.

3.4.4 NAMD

The forcefield files generated by SMOG v2 are fully compatible with NAMD. To perform SMOG models in NAMD, please consult the NAMD manual. A tutorial is available: http://vidar.scs.uiuc.edu/jlai7/Tutorial/GoDemo.pdf.

Chapter 4

Template-Based Approach

4.1 Introduction to templates

SMOG v2 offers increased versatility over SMOG v1 [9] by shifting to a template-based approach for defining molecular structures. Each template allows for direct control of the structure-based energy function, which may include (but is not limited to) multi-resolution models, and models that include non-specific interactions. The plug-and-play nature of the templates has the additional advantage of forcefield portability and easy sharing of user-created variations of structure-based potentials.

A single SMOG "template" is comprised of four XML-formatted files. These files are **absolutely necessary** when using SMOG v2. XML format was adopted because of its consistent formatting, ease of editability and readability, and there are widely available program modules to generate, and parse XML files. Furthermore, XML allows for schemas, a form content format restriction file, to which the template files must conform, which adds an additional layer of error checking capabilities. This chapter assumes that the user knows the basics of XML formatting. Users unfamiliar with XML formatting may want to check out the W3schools' website.

Table 4.1 summarizes the purpose of each template file. In Chapter 5, we show how to add new new residue to the template files for an All-Atom structure-based model.

4.2 SMOG v2 Templates

SMOG v2 expects four template files to be present in a single folder (i.e. the template folder). As discussed in Chapter 3, the template folder name is a required argument when running SMOG v2. A template folder can only contain **one** of each file type. If

File	Purpose
Biomolecular Information File (.bif)	Defines the structure of biomolecules to be supported
Setting Information File (.sif)	Defines interaction function declarations
Bond Information File (.b)	Defines bonded interactions between atoms
Nonbond Information File (.nb)	Defines non-bonded interactions between atoms

TABLE 4.1: Descriptions of the 4 files that comprise a single template. The expected suffix of each file is shown in parentheses.

your template folder contains more than one file of a specific file type, the program will exit with an error. Each file contains unique information, as described below.

4.2.1 Biomolecular Information File (.bif)

The Biomolecular Information File (from here on called .bif) defines the covalent structure of all residues described by a particular forcefield. The .bif file is used to extract the appropriate coordinate information from the PDB file. Since the PDB file only provides the coordinates, numbers and names of atoms and residues, the residue definitions in the .bif file are used in conjunction with the PDB file to build a forcefield for a particular biomolecule. Each residue is defined in the .bif file by declaring all the atoms in that residue, the bonds between the atoms and the improper dihedrals between the atoms.

Residues

Each residue is individually defined between the `<residues>` and `</residues>` tags. As an example, the text below shows how one would define the residue ALA, which contains 5 atoms.

```
1               <residue name="ALA" residueType="amino" atomCount="5">
2                       <atoms>
3                               <atom bType="B_1" nbType="NB_1" pairType="P_1">N</atom>
4                               <atom bType="B_1" nbType="NB_1" pairType="P_1">CA</atom>
5                               <atom bType="B_1" nbType="NB_1" pairType="P_1">C</atom>
6                               <atom bType="B_1" nbType="NB_1" pairType="P_1">O</atom>
7                               <atom bType="B_1" nbType="NB_1" pairType="P_1">CB</atom>
8                       </atoms>
9                       <bonds>
10      <!--BACKBONE-->
11                              <bond energyGroup="bb_a">
12                                      <atom>N</atom>
13                                      <atom>CA</atom>
14                              </bond>
15                              <bond energyGroup="bb_a">
```

```
16                                      <atom>CA</atom>
17                                      <atom>C</atom>
18                              </bond>
19                              <bond energyGroup="bb_a">
20                                      <atom>C</atom>
21                                      <atom>O</atom>
22                              </bond>
23          <!--FUNCTIONAL GROUP-->
24                              <bond energyGroup="sc_a">
25                                      <atom>CA</atom>
26                                      <atom>CB</atom>
27                              </bond>
28                      </bonds>
29                      <impropers>
30                              <improper>
31                                      <atom>CB</atom>
32                                      <atom>CA</atom>
33                                      <atom>C</atom>
34                                      <atom>N</atom>
35                              </improper>
36                      </impropers>
37              </residue>
```

LISTING 4.1: Residue section of .bif file

The attribute *name* is the name of the residue, as used in your PDB file. The attribute residue *residueType* is the type of residue, in this case, an amino residue. Finally the optional attribute *atomCount* allows the user to explicitly set the total number of atoms to be counted for normalizing energies. That is, the total atom count is used in the energetic scaling procedure of dihedrals and contact energies, as described in Appendix A. This feature is useful when including many copies of a ligand in your system, since the energetic normalization should only be based on the protein, or RNA, and not the multiply-copied ligands. In such a scenario, the user would set *atomCount* to 0. If *atomCount* is not defined, SMOG v2 automatically uses the total number of atoms listed under the `<atoms>` tag.

The `<atoms>` tag declares all the atoms in the residue. Note that all the atoms within a specific residue in your PDB **must be** listed here. If the PDB and .bif are not consistent, the program will terminate with an error. This is the reason that the default templates differentiate between the C-terminal and non-terminal protein residues (See Chapter 3). Each atom has a bond type *bType*, a non-bond type *nbType* and a pair type *pairType*. The bond type attribute is used in the generation of the bonded interactions (bonds, angles, and dihedrals). Likewise, the non-bond type attribute is used in the generation of the non-bonded interactions. The pairType attribute is used in the generation of contact interactions (6-12, 10-12 or Gaussian interactions).

The `<bonds>` tag contains all the bonds that should be present in the residue. Each bond in a residue is listed under the `<bond>` tag. The atom names must match those listed in the `<atoms>` field. The bond tag also has an attribute called *energyGroup* that allows for one to define heterogeneous energetics in the system. The energy group attribute is used in conjunction with the bond types to determine the dihedral interaction. Using the bonds declared here, the program dynamically calculates all angles and dihedrals that can exist in the molecule.

The `<impropers>` tag contains all the improper dihedral angles in the biomolecule. The tag `<improper></improper>` holds four atom tags. The order of the four atoms defines a specific improper dihedral within a residue. This feature is used to add dihedrals that cannot be determined based on bond geometry.

Connections

In addition to defining a residue, the .bif file is also used to define how sequential residues are covalently connected. Listing 4.2.1 shows how two residues of type amino are covalently linked. The attribute *residueType1* and *residueType2* declares how a residue of type residueType1 at position n should be connected to a residue of type residueType2 at position n+1. The residue types are matched based on the residue definitions. Much of the structure of the connection element is similar to that of the residue element. There is a single bond, whereby the first atom belongs to the nth residue and the second atom belongs to the (n+1)th residue. We can also define, though not a requirement component of all connection definitions, a single improper dihedral. In the context of impropers, the special character suffix "+" is used to declare atoms that belong to the (n+1)th residue. In the code listing, the N atom belongs to the (n+1)th residue.

```
1       <connections>
2               <!-- AMINO/AMINO -->
3                       <connection name="amino-amino" residueType1="amino" residueType2="amino">
4                       <bond energyGroup="r_a" >
5                               <atom>C</atom>
6                               <atom>N</atom>
7                       </bond>
8
9
10                      <improper>
11                              <atom>O</atom>
12                              <atom>CA</atom>
13                              <atom>C</atom>
14                              <atom>N+</atom>
15                      </improper>
16              </connection>
```

LISTING 4.2: Connection section of .bif file

4.2.2 Setting Information File (.sif)

While the .bif file is used to define the covalent geometry of each residue, the Setting Information File (.sif) is used to control the distribution and functional form of the energetics, which includes the inter-dihedral dihedral ratios, contact-to-dihedral ratios, contact map settings and function declarations.

Functions

The `<functions>` tag should list all the functions that the model requires.

```
<functions>
        <function name="bond_harmonic" directive="bonds"/>
        <function name="bond_type6" directive="bonds"/>
        <function name="angle_harmonic" directive="angles"/>
        <function name="angle_free" directive="angles"/>
        <function name="dihedral_cosine" directive="dihedrals"/>
        <function name="dihedral_harmonic" directive="dihedrals"/>
        <function name="dihedral_free" directive="dihedrals"/>
        <function name="contact_1" directive="pairs" exclusions="1"/>
        <function name="contact_2" directive="pairs" exclusions="1"/>
        <function name="contact_gaussian" directive="pairs" exclusions="1"/>
</functions>
```

LISTING 4.3: Example of functions section of a .sif file

Each function is defined using the `<functions>` tag. These function names are mapped to specific subroutines in `src/smogv2`. To add a new interaction type, follow the examples already in the code. Note: These functions are simply mappings to already defined Gromacs interactions. If it doesn't exist in Gromacs, it won't help to add it here. The interactions currently in the code are listed in Table 4.2. If you add useful interactions please let us know so that we can incorporate them into the default distribution.

Group Settings

The structure-based model has two classes of energy groups: contact groups, and dihedral groups. Each contact group represents a collection of contacts, and each dihedral

[1] Can be designated as a contact potential in .nb for applications like restraining bound ions or elastic network models.

[2] Minimum at native distance if using ? for c6/c12.

[3] Minimum at native distance if using ? for cN/cM. Need to include a table file with `-tablep` with `mdrun`

[4] Details in Appendix A.3.4.1.

Name	**.top**	**ftype**	**Input Parameters**
bond_harmonic	[bonds]	1	r_0, k
bond_type6[1]	[bonds]	6	r_0, k
angle_harmonic	[angles]	1	θ_0, k
dihedral_harmonic	[dihedrals]	2	ϕ_0, k
dihedral_cosine	[dihedrals]	1	ϕ_0, k, multiplicity
contact_1[2]	[pairs]	1	N, M, c6, c12
contact_2[3]	[pairs]	1	N, M, cN, cM
contact_gaussian[4]	[pairs]	6	ϵ_C, r_{NC} , σ, r_0

TABLE 4.2: Functions available in SMOG2.

group represents a collection of dihedrals. These energy groups are used for energetic scaling of interaction strength.

$$H_{AA} = ... + \sum_{backbone} \epsilon_{BB} F_D(\phi) + \sum_{sidechain} \epsilon_{SC} F_D(\phi) \tag{4.1}$$

$$+ \sum_{contacts} \epsilon_C F_{contacts}(r) + ... \tag{4.2}$$

Shown above is the All-Atom Hamiltonian with only the dihedral and contact terms. Shown below are the energetic scalings of the dihedral and contact terms, and their respective attributes under the .sif file.

$$\frac{\epsilon_{BB}}{\epsilon_{SC}} = \frac{\text{intraRelativeStrength bb}}{\text{intraRelativeStrength sc}} \tag{4.3}$$

$$\sum \epsilon_{BB} + \sum \epsilon_{SC} + \sum \epsilon_C = \text{Total non-ligand atoms} \tag{4.4}$$

$$\frac{\sum \epsilon_{BB} + \sum \epsilon_{SC}}{\sum \epsilon_C} = \frac{\text{dihedrals groupRatio}}{\text{contacts groupRatio}} \tag{4.5}$$

The program automatically calculates the total number of non-ligand atoms used in the energetic scaling. Although the scaling equations shown above is limited to residue types with only two dihedral types (backbone and sidechain dihedrals), and single contact type, the program allows for scaling equations to be generalized to more than two dihedral types and more than one contact type. The energy group ratios are contained within the `<Groups>` tag.

1 `<energyGroup name="bb_n" residueType="nucleic" intraRelativeStrength="1" normalize="1"/>`

```
2        <energyGroup name="sc_n" residueType="nucleic" intraRelativeStrength="1" normalize="1"/>
3        <energyGroup name="pr_n" residueType="nucleic" normalize="0"/>
4        <energyGroup name="ip_n" residueType="nucleic" normalize="0"/>
5        <energyGroup name="r_n"  residueType="nucleic" normalize="0"/>
6        <contactGroup name="c" intraRelativeStrength="1" normalize="1"/>
7        <groupRatios contacts="2" dihedrals="1"/>
```

LISTING 4.4: Energy group section of .sif file

The two classes of energy group ratios, dihedral and contact ratios, are controlled under the `<energyGroup>` and `<contactGroup>` tags respectively. The *residueType* attribute is used to designate the residue type the scaling factors of a particular energy group is used for. The *name* attribute is the label for the energy group. The *name* attribute used here is matched to the *energyGroup* attribute under `<bond>` tag in the .bif file. The name of a particular contact energy group will be used later when declaring contact interaction functions in the subsequent section of this chapter.

The *normalize* attribute for each energy group is a boolean attribute (1 or 0), and is used to determine if a particular energy group is to be included in energy normalization (see equation 4.4). For the All-Atom model, the dihedral group with the name "pr_n" (which represents the nucleic planar rigid dihedrals) has a normalize option set to 0, indicating that planar dihedral in nucleic acids will not be part of the normalization. In contrast, in the All-Atom model, sidechain dihedrals are normalized, as are backbone dihedrals and contact energies. Accordingly, those energy groups have the normalize option set to 1. The *intraRelativeStrength* attribute is the relative ratio of stabilizing energy within the different class of energy group for a particular residue (see equations 4.3).

Finally we use the `<groupRatios>` tag to set the energy partition between the two classes of energy group according to equation 4.5.

4.2.3 Bond Information File (.b) & Nonbond Information File (.nb)

The .b file is used to define all bonded interactions, which includes bonds, angles, and dihedral. The .nb file is used to define all the non-bonded interactions, such as contacts, 1-4 pairs, and excluded volume.

Bonded Interactions

We first discuss how to define a basic bond interaction for the All-atom model.

```
1  <!-- BONDS -->
2  <bonds>
3          <bond func="bond_harmonic(?,10000)">
4          <bType>*</bType>
5          <bType>*</bType>
6          </bond>
7          <bond func="bond_type6(?,200)">
8          <bType>*</bType>
9          <bType>MG</bType>
10         </bond>
11 </bonds>
```

LISTING 4.5: Bonds section of .b file

Recall that each atom is given a *bType* when they are declared in the .bif file. Given a particular bonded interaction (bonds, angles, dihedrals, impropers), the functional form for a bonded interaction is assigned by matching the combination of the *bType*s in that interaction. The first `<bond>` tag (line 3 in Listing 4.5) is used assign a function called bond_harmonic to two bonded atoms of any type. Since the vanilla model has only B_1 atoms, the *s could be replaced with B_1, and the same bonds would be assigned. Recall under Listing 4.3, line 2, we defined bond_harmonic() as a type 1 function under the bonds directive (harmonic bond function). The input parameters for bond_harmonic() are r_0 and ϵ_{bond}. In this case we use the special character "?" to tell SMOG to calculate the native bond length (r_0) from the PDB structure file. You can instead also give an empirical value for the bond distance. This feature is useful when adding nonspecific/empirical terms (e.g. an AMBER/CHARMM backbone) to the Hamiltonian.

The *bType* attribute here can take either an exact bond type or a special wildcard "*" character that matches to all available *bType*s. For the case of the All-Atom model, since all the *bType* is identical, we can instead also defined the bond interaction as shown under the `<bond>` tag in line 8 of Listing 4.5. **Please note that the program will assign the interaction that most closely matches the *bType*s of a given 4-atom pair.** One needs to be careful not to declare interactions that conflict with one another. For example, if your system contains a bond between atoms of bType B_1 and B_2, and bond definitions are only given for bTypes B_1-* and B_2-*, then SMOG would not know which function to apply, and it will exit with an error. However, if one also explicitly defined a bond function for B_1-B_2 pairs, that bond would take priority.

The angle interaction follow a similar form as bonds, but instead of expecting two *bType* attributes, it requires three. The *bType* attribute in this case is symmetric to the central *bType*. When matching bond angles, the angle definition that matches the most atoms

identically will be used. Again, if an equal number of atoms match in multiple angle definitions, there would be ambiguity, and SMOG will quit.

The "dihedral_cosine" function is classified under the dihedral directive with function type 1 in the .sif file (Listing 4.3 line 3). The arguments are of the form $(\phi_0, \epsilon_d, \text{mult})$. As introduced earlier the special character "?" tells the program to calculate the native value from the PDB structure file. In this case we ask the program to calculate the dihedral angle for all dihedral interactions that involve the *bType* combination *-*-*-*. By using the "?" argument, along with a multiplicity factor (third argument), we tell the program to multiply all the ϕ_0 values by the multiplicity factor. More generally, one may provide any function for the angle calculation. For example, if you were to use (?*2-1), the angle would be evaluated as: native angle, times 2 and minus 1.

$$F_D(\phi) = (1 - cos(\phi - \phi_0)) + 0.5(1 - cos(3(\phi - \phi_0))) \tag{4.6}$$

Equation 4.6 shown above is the dihedral interaction function used in the All-Atom model. The code Listing 4.6 shown below shows how to define the second term of $F_D(\phi)$

```
1 <dihedral func="dihedral_cosine(?,?,3)+dihedral_cosine(?,?,1)" energyGroup="bb_n">
2         <bType>*</bType>
3         <bType>*</bType>
4         <bType>*</bType>
5         <bType>*</bType>
6 </dihedral>
```

LISTING 4.6: Dihedral section of .b file

Finally multiple function can be applied to the same dihedral angle by including a sum of functions (e.g. func="f(..)+g(..)+h(..)").

The special keyword "?" has limitations in where it can be used. For bonds and angles, it can only be used for the first input parameter to a function. For dihedrals declared as type 1, it can be used for the first two input parameters to a function. In the case of dihedrals, the second input parameter for a type 1 dihedral is the energetic scaling term. The "?" option is used to tell the program that the dihedral is considered under the global energy normalization procedure. ϵ_d is dynamically calculated by the program.

A note on assigning dihedrals: As with bonds and angles, it is quite easy to provide multiple *bType* sequences that will match to the same atoms in a system. For example, if you define a dihedral function for B_1-B_2-*-*, as well as B_1-*-*-*, then a dihedral in your system between atoms (B_1,B_2,B_1,B_1) would match both. To determine which

function should be applied, a scoring function S is used. S is defined as 2 times the number of exactly matching *bTypes*, plus 1 times the number of wildcard matches. If any one of the atoms does not have an exact match, nor a wildcard, then $S = 0$. According to this criteria, $S = 6$ for the first definition and $S = 5$ for the second. Accordingly, the B_1-B_2-*-* definition would take priority. If two dihedrals have equal S values, $S = 0$ for all dihedral functions, then the program will quit with an error.

Non-Bonded Interactions

The .nb files is used to generate non-bonded interactions such as native contacts and non-specific interactions.

```
1  <!-- DEFAULTS -->
2  <defaults gen-pairs="0"/>
3  <!-- GENERAL NONBONDS -->
4  <nonbond mass="1.00"  charge="0.000" ptype="A"  c6="0.0"  c12="5.96046e-9">
5          <nbType>NB_1</nbType>
6  </nonbond>
7  <!-- CONTACTS -->
8  <contact func="contact_1(6,12,?,?)" contactGroup="c">
9          <pairType>*</pairType>
10         <pairType>*</pairType>
11 </contact>
```

Listing 4.7: Contacts section of .nb file

Listing 4.7 shows how to define non-bonded and native contact parameters. The *nbType* value defined for each atom in the .bif is matched to the nonbond declaration.

Note: the contacts are placed in the `[ pairs ]` section in the topology file which was originally used for 1-4 pair interactions. For structure based models, it has been applied to include native contacts.

For structure-based models, a non-bonded interaction is a volume exclusion interaction usually defined as the Lennard-Jones 12 term. Gromacs generates these interactions using the information provided in the `[ defaults ]` and the `[ atomtypes ]` sections. The non-bonded interaction declaration in SMOG2 contains the atom attributes: mass, charge and atom type, as well as the explicit c6 and c12 terms for the Lennard-Jones function. If one wanted to include non-specific attractive interactions between atoms, then a non-zero value should be given for the c6 parameter.

A contact function declaration includes the additional *contactGroup* attribute, which is used to map the function to specific groups of contacts. For example, in Listing 4.4, the contact group c was declared with *intraRelativeStrength*=1 and *normalize* option set to 1 (true).

SMOG2 supports two types of interactions for native contacts. Since the contact interactions are unique to SMOG v2, refer to Table 4.2 for how SMOG2 expects the input parameters. In code Listing 4.7, the contact parameters ϵ_C, c6, and c12 are calculated automatically by the program using energy normalization and the PDB structure because of the 3 ?. ϵ_C is also calculated dynamically through the scaling equations. If one uses the 10-12 or N-M functions, table files have to be included (see Section 3.4.1.2 for syntax and details.)

Chapter 5

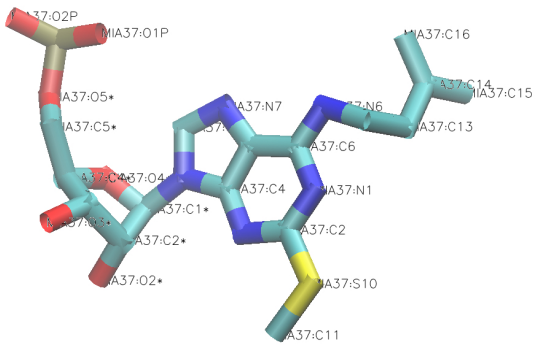
Adding a new residue

This chapter provides a step-by-step tutorial on how to add a new residue type using SMOG v2 template files.

5.1 Step 1 - Examine the molecular structure

The residue 2-methylthio-N6 isopentenyl adenosine (MIA) is a modified nucleic acid residue that is present in many RNA structures. It is identical to Adenine, except there are a few additional carbon atoms and a sulfur atom ligated to XXX. For the purposes of demonstrating We would also like to define a larger mass for sulfur atom.

Make sure to have the correct chemical structure of your molecule. A useful method is to visually inspect it with a molecular visualization program (VMD for example):



5.2 Step 2 - Create a new All-Atom template directory

Since we're going to explicitly define each atom in the MIA residue, the model used here will be All-Atom model. You can modify the default All-Atom directory provided by Smog2 - `SBM_AA` or create a copy of it in your working directory to make the changes in the template files. In this example we will modify the existing `AA-whitford09.bif` and `AA-whitford09.nb` files.

5.3 Step 3 - Define a new residue

As mentioned in chapter 4, the biomolecular information file (.bif) declares the structure of all biomolecules in your system. Here we will define the residue information by declaring all of the atoms, bonds and improper dihedrals within the residue. The full modified file can be found in the `XMLcode_examples/AA-whitford09_withMIA.bif` directory.

5.3.1 Place the new residue tag in the .bif file

As you get acquainted with the `AA-whitford09.bif` file structure, you'll find that the residues appear grouped together according to their type: Ligands, amino and nucleic residues. The residue type of MIA is nucleic, so it is added for convenience next to the existing nucleic residues. The `<residue>` tag encapsulates all of the residue information.

```
2945        <!-- NUCLEIC RESIDUES -->
2946
2947  <!--2-methylthio-N6 isopentenyl adenosine-->
2948                <residue name="MIA" residueType="nucleic">
2949                        <atoms>
2950                        </atoms>
2951                        <bonds>
2952                        </bonds>
2953                        <impropers>
2954                        </impropers>
2955                </residue>
2956
2957  <!--RNA A-->
2958                <residue name="A" residueType="nucleic">
```

LISTING 5.1: Nucleic residue section of .bif file

Keep in mind that the attribute *name* should match the residue name in the PDB file.

5.3.2 List all of the atoms in the residue

List all of the atom names in your residue as they appear in your PDB. The `<atoms>` tag encapsulates all the atoms in the biomolecule.

```
2945        <!-- NUCLEIC RESIDUES -->
2946
2947  <!--2-methylthio-N6 isopentenyl adenosine-->
2948                <residue name="MIA" residueType="nucleic">
2949                        <atoms>
2950                                <atom bType="B_1" nbType="NB_1" pairType="P_1">P</atom>
2951                                <atom bType="B_1" nbType="NB_1" pairType="P_1">O1P</atom>
2952                                <atom bType="B_1" nbType="NB_1" pairType="P_1">O2P</atom>
2953                                <atom bType="B_1" nbType="NB_1" pairType="P_1">O5*</atom>
2954                                <atom bType="B_1" nbType="NB_1" pairType="P_1">C5*</atom>
2955                                <atom bType="B_1" nbType="NB_1" pairType="P_1">C4*</atom>
2956                                <atom bType="B_1" nbType="NB_1" pairType="P_1">O4*</atom>
2957                                <atom bType="B_1" nbType="NB_1" pairType="P_1">C3*</atom>
2958                                <atom bType="B_1" nbType="NB_1" pairType="P_1">O3*</atom>
2959                                <atom bType="B_1" nbType="NB_1" pairType="P_1">C2*</atom>
2960                                <atom bType="B_1" nbType="NB_1" pairType="P_1">O2*</atom>
2961                                <atom bType="B_1" nbType="NB_1" pairType="P_1">C1*</atom>
2962                                <atom bType="B_1" nbType="NB_1" pairType="P_1">N9</atom>
2963                                <atom bType="B_1" nbType="NB_1" pairType="P_1">C8</atom>
2964                                <atom bType="B_1" nbType="NB_1" pairType="P_1">N7</atom>
2965                                <atom bType="B_1" nbType="NB_1" pairType="P_1">C5</atom>
2966                                <atom bType="B_1" nbType="NB_1" pairType="P_1">C6</atom>
2967                                <atom bType="B_1" nbType="NB_1" pairType="P_1">N6</atom>
2968                                <atom bType="B_1" nbType="NB_1" pairType="P_1">N1</atom>
2969                                <atom bType="B_1" nbType="NB_1" pairType="P_1">C2</atom>
2970                                <atom bType="B_1" nbType="NB_1" pairType="P_1">N3</atom>
2971                                <atom bType="B_1" nbType="NB_1" pairType="P_1">C4</atom>
2972                                <atom bType="B_1" nbType="NB_2" pairType="P_1">S10</atom>
2973                                <atom bType="B_1" nbType="NB_1" pairType="P_1">C11</atom>
2974                                <atom bType="B_1" nbType="NB_1" pairType="P_1">C12</atom>
2975                                <atom bType="B_1" nbType="NB_1" pairType="P_1">C13</atom>
2976                                <atom bType="B_1" nbType="NB_1" pairType="P_1">C14</atom>
2977                                <atom bType="B_1" nbType="NB_1" pairType="P_1">C15</atom>
2978                                <atom bType="B_1" nbType="NB_1" pairType="P_1">C16</atom>
2979                        </atoms>
2980                        <bonds>
2981                        </bonds>
2982                        <impropers>
2983                        </impropers>
2984                </residue>
2985
2986  <!--RNA A-->
```

LISTING 5.2: Adding the atoms section to the residue structure

In this example, as in the default models, all bonded interactions (bonds, angles and dihedrals) are defined the same for all atoms. Therefore, only one atom group needs to be defined. The bond type:*bType*=`B_1` is identical for all atoms. The contact interactions

pairType=P_1 are also defined to be the same for all atoms. However, changing the mass of a specific atom, such as sulfur (S10) in our example will produce a different non-bonded interaction due to a different volume exclusion term. The new group type of *nbType*=NB_2 will be defined in the .nb file.

5.3.3 List all of the atom bonds

The chemical structure should tell you how atoms are connected to each other via chemical bonds. Inspect those bonds and add them to the <bonds> section. The <bonds> tag encapsulates all the bonds in the biomolecule.

```
                    <bonds>
                <!--BACKBONE-->
                                <bond energyGroup="bb_n">
                                        <atom>P</atom>
                                        <atom>O1P</atom>
                                </bond>
                                <bond energyGroup="bb_n">
                                        <atom>P</atom>
                                        <atom>O2P</atom>
                                </bond>
                                <bond energyGroup="bb_n">
                                        <atom>P</atom>
                                        <atom>O5*</atom>
                                </bond>
                                <bond energyGroup="bb_n">
                                        <atom>O5*</atom>
                                        <atom>C5*</atom>
                                </bond>
                                <bond energyGroup="bb_n">
                                        <atom>C5*</atom>
                                        <atom>C4*</atom>
                                </bond>
                                <bond energyGroup="bb_n">
                                        <atom>C4*</atom>
                                        <atom>O4*</atom>
                                </bond>
                                <bond energyGroup="bb_n">
                                        <atom>C4*</atom>
                                        <atom>C3*</atom>
                                </bond>
                                <bond energyGroup="bb_n">
                                        <atom>C3*</atom>
                                        <atom>O3*</atom>
                                </bond>
                                <bond energyGroup="bb_n">
                                        <atom>C3*</atom>
                                        <atom>C2*</atom>
                                </bond>
                                <bond energyGroup="bb_n">
                                        <atom>C2*</atom>
```

```
3020                                    <atom>O2*</atom>
3021                            </bond>
3022                            <bond energyGroup="bb_n">
3023                                    <atom>C2*</atom>
3024                                    <atom>C1*</atom>
3025                            </bond>
3026                            <bond energyGroup="bb_n">
3027                                    <atom>C1*</atom>
3028                                    <atom>O4*</atom>
3029                            </bond>
3030                    <!--FUNCTIONAL GROUP-->
3031                                    <bond energyGroup="sc_n">
3032                                            <atom>C1*</atom>
3033                                            <atom>N9</atom>
3034                                    </bond>
3035                                    <bond energyGroup="pr_n">
3036                                            <atom>N9</atom>
3037                                            <atom>C8</atom>
3038                                    </bond>
3039                                    <bond energyGroup="pr_n">
3040                                            <atom>C8</atom>
3041                                            <atom>N7</atom>
3042                                    </bond>
3043                                    <bond energyGroup="pr_n">
3044                                            <atom>N7</atom>
3045                                            <atom>C5</atom>
3046                                    </bond>
3047                                    <bond energyGroup="pr_n">
3048                                            <atom>C5</atom>
3049                                            <atom>C6</atom>
3050                                    </bond>
3051                                    <bond energyGroup="pr_n">
3052                                            <atom>C6</atom>
3053                                            <atom>N6</atom>
3054                                    </bond>
3055                                    <bond energyGroup="pr_n">
3056                                            <atom>C6</atom>
3057                                            <atom>N1</atom>
3058                                    </bond>
3059                                    <bond energyGroup="pr_n">
3060                                            <atom>N1</atom>
3061                                            <atom>C2</atom>
3062                                    </bond>
3063                                    <bond energyGroup="pr_n">
3064                                            <atom>C2</atom>
3065                                            <atom>N3</atom>
3066                                    </bond>
3067                                    <bond energyGroup="pr_n">
3068                                            <atom>N3</atom>
3069                                            <atom>C4</atom>
3070                                    </bond>
3071                                    <bond energyGroup="pr_n">
3072                                            <atom>C4</atom>
3073                                            <atom>C5</atom>
3074                                    </bond>
```

```
                                <bond energyGroup="pr_n">
                                        <atom>N9</atom>
                                        <atom>C4</atom>
                                </bond>
        <!--ADDITIONAL TO "A" FUNCTIONAL GROUP-->
                                <bond energyGroup="pr_n">
                                        <atom>C2</atom>
                                        <atom>S10</atom>
                                </bond>
                                <bond energyGroup="pr_n">
                                        <atom>S10</atom>
                                        <atom>C11</atom>
                                </bond>
                                <bond energyGroup="pr_n">
                                        <atom>N6</atom>
                                        <atom>C12</atom>
                                </bond>
                                <bond energyGroup="pr_n">
                                        <atom>C12</atom>
                                        <atom>C13</atom>
                                </bond>
                                <bond energyGroup="pr_n">
                                        <atom>C13</atom>
                                        <atom>C14</atom>
                                </bond>
                                <bond energyGroup="pr_n">
                                        <atom>C14</atom>
                                        <atom>C15</atom>
                                </bond>
                                <bond energyGroup="pr_n">
                                        <atom>C14</atom>
                                        <atom>C16</atom>
                                </bond>
                        </bonds>
```

Listing 5.3: Adding the bonds section to the residue structure

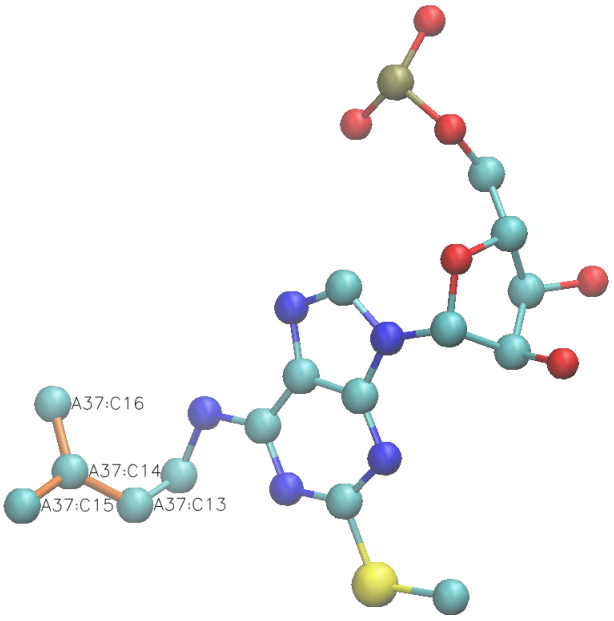
Note that the bonds are separated by the comments: `BACKBONE,FUNCTIONAL GROUP, ADDITIONAL BONDS TO 'RNA A' FUNCTIONAL GROUP`. The bond tag attribute `energyGroup` classifies the energy group the specific bond belongs to and helps to determine the dihedral strengths. The energy group `bb_n` refers to bonds that belong to the backbone group, and `pr_n` refers to the functional group. It is sometimes useful to use an existing residue as a reference if we know that the new residue is a slight modification of it. In our example, MIA is a modified RNA A molecule. All backbone and functional group bonds of RNA A can be added to MIA and we are left to determine the few other bonds created by the additional atoms: S10, C11-C16. The additions are added under the comment `ADDITIONAL BONDS TO 'RNA A' FUNCTIONAL GROUP` with energy group of `pr_n`.

5.3.4 List the improper dihedrals

Improper dihedrals cannot be dynamically calculated by the program using the bonds, and should be added separately. The tag `<improper></improper>` holds four atoms. The order of the four atoms here defines a specific improper dihedral within a biomolecule.

How to find an improper dihedral:

An improper dihedral is used to ensure proper geometry about a chiral centers (i.e. prevent symmetry inversion due to an absent hydrogen atom). A proper dihedral would be defined by four sequential atoms connected by bonds (such as the dihedral C4*-C5*-O5*-P). An improper dihedral is defined by atoms that are not sequential. Those angles need to be identified using the chemical structure of the molecule. For example we consider the four carbon atoms: C13,C14,C15,C16 and their bonds as defined above (highlighted in orange).



Finally, we add all of the improper dihedrals to our residue. In our example there is only one additional improper dihedral described above.

Add the improper dihedrals section to the residue structure:

```
3109                    <impropers>
```

```
3110                                    <improper>
3111                                            <atom>C3*</atom>
3112                                            <atom>C4*</atom>
3113                                            <atom>C5*</atom>
3114                                            <atom>O4*</atom>
3115                                    </improper>
3116                                    <improper>
3117                                            <atom>O3*</atom>
3118                                            <atom>C3*</atom>
3119                                            <atom>C4*</atom>
3120                                            <atom>C2*</atom>
3121                                    </improper>
3122                                    <improper>
3123                                            <atom>O2*</atom>
3124                                            <atom>C2*</atom>
3125                                            <atom>C1*</atom>
3126                                            <atom>C3*</atom>
3127                                    </improper>
3128                                    <improper>
3129                                            <atom>C2*</atom>
3130                                            <atom>C1*</atom>
3131                                            <atom>O4*</atom>
3132                                            <atom>N9</atom>
3133                                    </improper>
3134         <!--ADDITIONAL IMPROPER DIHEDRAL TO "RNA A" -->
3135                                    <improper>
3136                                            <atom>C13</atom>
3137                                            <atom>C14</atom>
3138                                            <atom>C15</atom>
3139                                            <atom>C16</atom>
3140                                    </improper>
3141                            </impropers>
3142                 </residue>
```

LISTING 5.4: Adding the improper dihedrals section to the residue structure

5.4 Step 4 - Define a non-bonded group in the .nb file

Adding a new atom of a different mass requires a creation of a new non-bonded group for the excluded volume interactions. In our example we chose the atom sulfur to have larger mass of twice the carbon mass. A new `<nonbond>` tag is added and it encapsulates the new non-bond group information such as mass charge and other explicit non-bonded terms. The mass is doubled in the `mass` entry. The *nbType* includes the previously defined group name `NB_2` that is consistent with the .bif file. The modified file can be found in: `XMLcode_examples/AA-whitford09_withMIA.nb`.

Adding a new non-bonded group in the .nb fie:

```
<?xml version='1.0'?>
<nb>
 <!-- DEFAULTS -->
 <defaults gen-pairs="0"/>
 <!-- GENERAL NONBONDS -->
  <nonbond mass="2.00"  charge="0.000" ptype="A"  c6="0.0"  c12="5.96046e-9">
    <nbType>NB_2</nbType>
 </nonbond>
 <nonbond mass="1.00"  charge="0.000" ptype="A"  c6="0.0"  c12="5.96046e-9">
    <nbType>NB_1</nbType>
 </nonbond>
<!-- CONTACTS -->
<contact func="contact_1(6,12,?,?)" contactGroup="c">
        <pairType>*</pairType>
        <pairType>*</pairType>
</contact>
</nb>
```

Listing 5.5: Defining a new non-bonded group in the .nb file

Chapter 6

Additional supported options

As described above, it is the aim of SMOG2 that the user will be able to extend the models in a wide range of ways. Accordingly, it is not possible that we provide descriptions of every possible variation that one may explore. However, here, we make an effort to provide some examples of how to implement specific features that we think may be frequently of interest.

6.1 Adding specific bonds

There are often cases where the covalent geometry of the system can not be determined by a general set of rules. For example, disulfide bonds may be formed, or broken, depending on the oxidation state. As another example, sugar structures often have branching patterns, and therefore do not form linear chains. For these types of chemical bonds, we provide the `BOND` option in the PDB file. If you would like to add a chemical bond, then add `BOND` lines immediately after the END line in the PDB file. The formatting is the following:

```
BOND ChainIndex1 AtomIndex1 ChainIndex2 AtomIndex2 energygroup
```

ChainIndex1 and AtomIndex1 indicate the first atom involved in the bond. ChainIndex1 is the index of the chain in which the atom exists, starting at 1. AtomIndex1 is the number of the atom, *as it appears in the PDB file*. Note, ChainIndex1 is not necessarily the chain ID. ChainIndex2 and AtomIndex2 indicate the second atom involved in the bond. energygroup indicates the properties of any dihedral angles that have the new bond as a middle bond (See Chapter 4 for discussion on energy groups). For example, the following line:

```
BOND 1 51 4 100 r_p
```

would add a chemical bond between the atom numbered 51 in the first chain, and the atom numbered 100 in the 4th chain. The bond properties would be determined based on the .b file settings, and the energy group of any associated dihedrals would be r_p. When SMOG2 runs, it will write information to the screen as the `BOND` lines are detected. It will also write out information about what it interpreted the lines to mean, so you can verify that the intended bonds are being added.

6.2 Adding electrostatics and non-standard contact potentials.

While not part of the standard structure-based model, there are often times where it is desirable to add some degree of electrostatic interactions, or one would like to use interactions that are not of the 6-12 form.

If you would like to add charge to your system, then you will need to define atom types with varying charges, which is accomplished by modifying the .nb file (See Chapter 4). By default, Gromacs will treat these interactions as purely Coulombic. If you would like to use a screened electrostatic interaction (i.e. Debye-Huckle), then you need to supply a table. We supply a tool that will generate a screened-electrostatic look-up table, as implemented by Givaty and Levy[10]. To generate this table, you can use the tool `smog_tablegen`. This is the same tool used for non-standard N-M potentials (e.g. the 10-12 potential used for C_α models. See Chapter 3). Usage:

```
$SMOG_PATH/bin/smog_tablegen <M> <N> <ion conc.> <elec.  switch dist.> \
                    <elec.  truncate dist.> <table length> <output name>
```

`<ion conc.>` is the monovalent ion concentration, which sets κ. In order to avoid numerical issues by imposing a cutoff distance, `<elec.  switch dist.>` and `<elec. truncate dist.>` should be used. These will modify the potential, starting at a distance `<elec.  switch dist.>`, using a 4th-order polynomial, which will ensure that the potential smoothly switches to zero at `<elec.  truncate dist.>`. Caution: When using electrostatics, you will need to adjust your .mdp so that cutoffs are properly imposed during the simulation. The above flags only ensure that the table is correct.

Appendix A

Energetic Description of the Distributed Models

A.1 The All-atom model

All non-hydrogen atoms are explicitly represented, and the provided structure (file.pdb) is defined as the global potential energy minimum. Here, we provide a complete description of the default all-atom structure-based model energy function, which is defined by the template SBM_AA). All calculations employ reduced units. Each atom is represented as a single bead of unit mass, and the charge of each atom is set to zero. Covalent geometry is maintained through harmonic interactions that ensure the bond lengths, bond angles, improper dihedral angles and planar dihedral angles remain about the values found in file.pdb. Non-bonded atom pairs that are in contact in the provided structure between residues i and j, where $i > j + 3$ for proteins and $i \neq j$ for RNA, are given an attractive 6-12 potential. The minimum of each 6-12 interaction is set to the distance of that atom pair in the provided structure. All non-native interactions between atoms that are not in contact in file.pdb are repulsive. Contacts were defined according to the Shadow algorithm (See Appendix B, with an all-atom cutoff distance of 6 Å and a shadowing radius of 1 Å. The functional form of the potential is,

$$\begin{aligned} V = & \sum_{bonds} \epsilon_r (r_i - r_{i,o})^2 + \sum_{angles} \epsilon_\theta (\theta_i - \theta_{i,o})^2 + \\ & \textstyle\sum_{impropers} \epsilon_{\chi_{imp}} (\chi_i - \chi_{i,o})^2 + \sum_{planar} \epsilon_{\chi_{planar}} (\chi_i - \chi_{i,o})^2 \\ & + \sum_{backbone} \epsilon_{BB} F_D(\phi_i) + \sum_{sidechains} \epsilon_{SC} F_D(\phi_i) \\ & + \sum_{contacts} \epsilon_C [\left(\frac{\sigma_{ij}}{r}\right)^{12} - 2\left(\frac{\sigma_{ij}}{r}\right)^{6}] + \sum_{non-contacts} \epsilon_{NC} \left(\frac{\sigma_{NC}}{r_{ij}}\right)^{12} \end{aligned} \tag{A.1}$$

where,

$$F_D(\phi) = [1 - \cos(\phi_i - \phi_{i,o})] + \frac{1}{2}[1 - \cos(3(\phi_i - \phi_{i,o}))] \tag{A.2}$$

When using SMOG v2, all values may be adjusted by the user, such as defining stabilizing non-native interactions and including non-specific dihedral angles. However, for the default model $r_{i,o}$, $\theta_{i,o}$, $\chi_{i,o}$, $\phi_{i,o}$ and σ_{ij} are given the values defined by the provided structure. For the default model, the parameters are set to the following values:

$\epsilon_{\mathrm{r}} = 50\epsilon_0$, $\epsilon_\theta = 40\epsilon_0$, $\epsilon_{\chi_{imp}} = 10\epsilon_0$, $\epsilon_{\chi_{planar}} = 40\epsilon_0$, $\epsilon_{\mathrm{NC}} = 0.1\epsilon_0$, $\sigma_{\mathrm{NC}} = 2.5$Å, $\epsilon_0 = 1$.

Note that, relative to the original implementation of this model, ϵ_r is decreased by a factor of two, and ϵ_θ is increased. As discussed in the SMOG v2 publication (in preparation), this allows for a longer timestep of 0.002 to be utilized, which is larger than the originally-implemented 0.0005. When assigning dihedral interaction weights (ϵ_{BB} and ϵ_{SC}), dihedrals are first grouped if they have a common middle bond. For example, in a protein backbone, there are up to four dihedral angles that may be defined that have the $C - C_\alpha$ bond as the middle bond. Each dihedral group is given a summed weight of ϵ_{BB}, or ϵ_{SC}. The ratio $R_{BB/SC} = \frac{\epsilon_{BB}}{\epsilon_{SC}}$ is set to 1 for nucleic acid dihedral angles and and 2 for protein dihedral angles. ϵ_{BB} for protein and nucleic acids are equal. Dihedral strengths and contact strengths are scaled such that:

$R_{C/D} = \frac{\sum \epsilon_C}{\sum \epsilon_{BB} + \sum \epsilon_{SC}} = 2$

$\sum \epsilon_C + \sum \epsilon_{BB} + \sum \epsilon_{SC} = N\epsilon_0$

The sums are over all dihedral angles in the system, and N is the number of atoms in the system.

We are going to add a description here for where to look in the template files for each of these parameters. As we continue to work on the manual, these details will be provided.

A.2 The C_α model

The C_α model coarse-grains the protein as single bead of unit mass per residue located at the position of the α-carbon. $\vec{\mathbf{x_0}}$ denotes the coordinates of the native state and any subscript 0 signifies a value taken from the native state. The potential is given by

$$V_{C\alpha}(\vec{\mathbf{x}}, \vec{\mathbf{x_0}}) = \sum_{\text{bonds}} \epsilon_r (r - r_0)^2 + \sum_{\text{angles}} \epsilon_\theta (\theta - \theta_0)^2 + \sum_{\text{dihedrals}} \epsilon_{\text{D}} F_{\text{D}}(\phi - \phi_0) + \sum_{\text{contacts}} \epsilon_{\text{C}} \left[5 \left(\frac{\sigma_{ij}}{r} \right)^{12} - 6 \left(\frac{\sigma_{ij}}{r} \right)^{10} \right] + \sum_{\text{non-contacts}} \epsilon_{\text{NC}} \left(\frac{\sigma_{\text{NC}}}{r_{ij}} \right)^{12} \quad \text{(A.3)}$$

where the dihedral potential F_{D} is,

$$F_{\text{D}}(\phi) = [1 - \cos(\phi)] + \frac{1}{2}[1 - \cos(3\phi)]. \quad \text{(A.4)}$$

The coordinates $\vec{\mathbf{x}}$ describe a configuration of the α-carbons, with the bond lengths to nearest neighbors r, three body angles θ, four body dihedrals ϕ, and distance between atoms i and j given by r_{ij}. Protein contacts that are separated by less than 3 residues are neglected. Excluded volume is maintained by a hard wall interaction giving the residues an apparent radius of $\sigma_{\text{NC}} = 4$ Å. The native bias is provided by using the parameters from the native state $\vec{\mathbf{x_0}}$. Setting the energy scale $\epsilon \equiv k_{\text{B}}$, the coefficients are given the homogeneous values: $\epsilon_r = 100\epsilon$, $\epsilon_\theta = 20\epsilon$, $\epsilon_{\text{D}} = \epsilon_{\text{C}} = \epsilon_{\text{NC}} = \epsilon$.

A.3 Gaussian contact potential (+gaussian templates)

The Gaussian-shaped contact potentials (A.1) are available in the SMOG version of Gromacs and NAMD (see section A.3.3). These potentials are used when one desires control over either the shape of the excluded volume or the width of the attractive potential. They are also useful if contacts require minima in two places. In depth characterization of the Gaussian potentials with all-atom structure-based models using SMOG can be found in [11] (templates/SBM_AA+gaussian). They are explored in the context of a multi-basin C_α model here [12] (templates/SBM_calpha+gaussian).

A.3.1 templates/SBM_AA+gaussian

Changes the contact potential to `ftype = 6`, ϵ_{C}=?, r_0=?, σ=$\sqrt{(}\mu^2/(50 \ln 2))$,$r_{\text{NC}}$=same as normal excluded volume (ad defined by c12, such that *i.e.*, $\sigma_{\text{NC}} * \epsilon_{\text{NC}}^{1/12}$). The rather complex definition of the width of the Gaussian well σ is designed to model the variable width of the LJ potential. $C_{\text{AA}}(1.2r_0^{ij}, r_0^{ij}) \sim -1/2$ so σ is defined such that $G(1.2r_0^{ij}, r_0^{ij}) = -1/2$ giving $\sigma^2 = (r_0^{ij})^2/(50 \ln 2)$.

A.3.2 templates/SBM_calpha+gaussian

Changes the contact potential to `ftype = 6`, ϵ_C=?, r_0=?, σ=0.05 nm, r_{NC}=$(0.4)^{12}$.

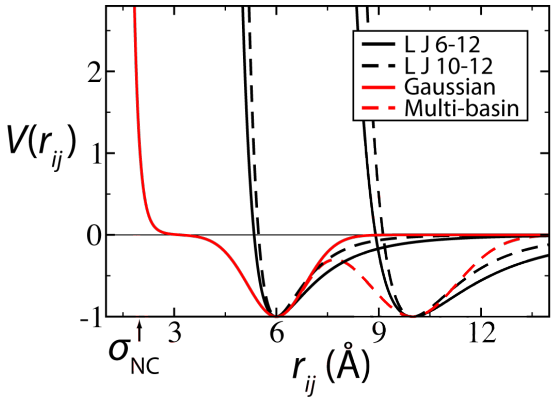


FIGURE A.1: Comparison of Lennard-Jones and Gaussian contact potentials. Black curves show LJ contact potentials with minima at 6 Å and 10 Å. The Gaussian contact potential shown in green has an excluded volume σ_{NC} that can be set independently of the location of the minimum. The dotted green line shows how the Gaussian contact would change as another minimum at 10 Å is added. Taken from [1].

A.3.3 Downloading the source code extensions

A.3.3.1 Gromacs

The Gaussian contact shapes are not available in the standard Gromacs distributions. The necessary source code can be obtained at http://smog-server.org/extension.html. This source distribution is compiled exactly as any other Gromacs source distribution.

A.3.3.2 NAMD

Currently the "nightly build" version of NAMD contains the Gaussian potentials in the "Go potentials" section. More information can be found in the NAMD manual.

A.3.4 Including Gaussian potentials in the topology files

A.3.4.1 Gromacs

The Gaussian interaction is designated in the `[ pairs ]` section of the topology file.

- `ftype = 6`

- $C_{ij} = -A\left(\left(1+\frac{1}{A}\frac{a}{r^{12}}\right)\left(1-\exp\left[-\frac{(r_{ij}-\mu_{ij})^2}{2\sigma_{ij}^2}\right]\right)-1\right)$
- $\epsilon_{\mathrm{C}} \rightarrow$ depth of the attractive well
- $r_0 \rightarrow$ position of the minimum of the attractive well
- $\sigma \rightarrow$ width of the attractive well
- $r_{\mathrm{NC}} \rightarrow$ position of the excluded volume hard wall
- This form includes an excluded volume part, and therefore the pair ij should be included in `[ exclusions ]`. The multiplicative form anchors the minimum of the well at $(r_0, -\epsilon_{\mathrm{C}})$.

Note that `ftype = 5` and `ftype = 7` exist in the SBM extensions version, though there is no implementation in SMOG2 at the moment. They can be added by hand if desired.

A.3.4.2 NAMD

Currently the "nightly build" version of NAMD contains the Gaussian potentials in the "Go potentials" section. More information can be found in the NAMD manual.

Appendix B

Understanding the provided SCM.jar tool

B.1 Introduction

This section describes a Java application `SCM.jar` that computes the "Shadow" map, a general contact definition for capturing the dynamics of biomolecular folding and function. It is described in the literature here [11]. A contact map is a binary symmetric matrix that encodes which atom pairs are given attractive interactions in the SBM potential. In the context of a SBM, the native contact map should approximate the distribution of stabilizing enthalpy in the native state that is provided by short range interactions like van der Waals forces, hydrogen bonding, and salt bridges. Any long range interactions or nonlocal effects are taken into account in a mean field way through the native bias.

B.1.1 Role of `SCM.jar` in SMOG v2

Internally SMOG v2 uses `SCM.jar` to compute contact maps. From the user's point of view the contact map can be of two types, all-atom or coarse-grained. An all-atom map returns the atoms that are in contact based on the Shadow definition. A coarse-grained map (e.g. to be used with the Calpha model) is created from an all-atom map. The coarse-grained map consists of residue-level contacts. A residue-level contact exists if there is at least one atom-atom contact between two residues. This is why a PDB containing all heavy atoms is required by the tool. When coarse-graining SMOG2 asks that the user provide an all-atom template in addition to the coarse-graining template that tells SMOG2 how to interpret the all-atom PDB in order to interface with `SCM.jar`.

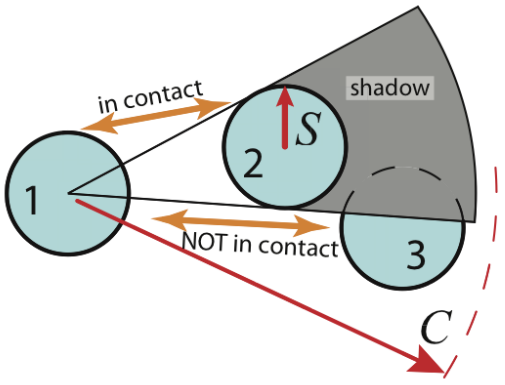


FIGURE B.1: The Shadow contact map screening geometry. Only atoms within the cutoff distance C are considered. Atoms 1 and 2 are in contact because they are within C and have no intervening atom. To check if atoms 1 and 3 are in contact, one checks if atom 2 shadows atom 1 from atom 3. The three atoms are viewed in the plane, and all atoms are given the same shadowing radius S. Since a light shining from the center of atom 1 causes a shadow to be cast on atom 3, atoms 1 and 3 are not in contact.

The tool is available with the SMOG2 source for users that want to create their own customized maps. The rest of the chapter describes the basics of using the tool.

B.1.1.1 Some details of coarse-graining

The coarse-grained contact map returned is only strictly recommended for use with Calpha models of proteins, and where the input PDB has an all-atom representation. For various modeling applications, it is desirable that the program not die with an error if the PDB doesn't only contain all-atom protein with each residue containing a CA atom. Therefore, the behavior is that the program will choose one atom from each residue to stand in as the representative coarse-grained position. It chooses, in order of preference: `CA`, `N1`, first atom in the residue. This really only matters for the `--distance` option.

B.1.2 Locating `SCM.jar`

The jar should be located in $SMOG_PATH/tools.

B.1.3 Citing `SCM.jar`

The citation for `SCM.jar` is [11].

B.1.4 Running `SCM.jar`

Like any java application, no compilation is necessary, but a virtual machine is required; `SCM.jar` requires a sufficiently recent JRE. `SCM.jar` reads SMOG formatted Gromacs input files. **Important! The all (heavy) atom geometry must be used**, even if the output will be a coarse-grained residue-based map for a C_α model. The atomic coordinates are read in `.gro` format and the bond connectivity is read via a `.top` obtained from the SMOG webtool (or source distribution). The topology is required since bonded atoms shadow each other differently and since contacts are automatically discarded between two atoms if they share a bonded interaction. At the command line, the basic syntax is

```
user$ java [-Xmx1000m] -jar SCM.jar -g grofile -t topfile -o outputName
[--chain chainFile] [--default | -m {shadow,cutoff}]
```

`-Xmx1000m` assigns 1000 MB of RAM to the Java virtual machine heap. With large complexes (>1e5 atoms) the default heap allocation can run out which gives the following error:

```
java.lang.OutOfMemoryError: Java heap space
```

The output all-atom contact file format is

```
chain_i atom_i chain_j atom_j [distance]
```

and similarly, the output residue contact file format is

```
chain_i residue_i chain_j residue_j [distance]
```

B.1.4.1 Some examples

- Shadow map, atomic contacts, shadowing radius 1 Å and cutoff 6 Å (default sizes). See Figure B.1 for definition of radius and cutoff. Add `--chain` if you have multiple chains, since the `.gro` format does not allow for chain information. Specify the chains file you get from your SMOG output.

  ```
  user$ java -jar SCM.jar -g protein.gro -t protein.top -o contactsOut
  --default [--chain chainsFile]
  ```

- Shadow map, atomic contacts, shadowing radius 2 Å and cutoff 4 Å.

```
user$ java -jar SCM.jar -g protein.gro -t protein.top -o contactsOut
-m shadow -c 4 -s 2 [--chain chainsFile]
```

- Cutoff map, atomic contacts, and cutoff 4 Å.

```
user$ java -jar SCM.jar -g protein.gro -t protein.top -o contactsOut
-m cutoff -c 4 [--chain chainsFile]

- OR -

user$ java -jar SCM.jar -g protein.gro -t protein.top -o contactsOut
-s shadow -s 0 -c 4 [--chain chainsFile]
```

- Shadow map, residue contacts, default, include contact distances

```
user$ java -jar SCM.jar -g protein.gro -t protein.top -o contactsOut
   --distances --coarse CA [--chain chainsFile]
```

- To calculate over a trajectory instead of a single structure, use `--multiple X`, where `X` is the number of frames in the trajectory `.gro` file. Assumes that the format of `proteinTraj.gro` is the same as the output of `trjconv`. This saves time relative to looping over many grofiles because the topology (and therefore the bonded list) is only created once.

```
user$ $GROMACS/trjconv -f traj.xtc -o proteinTraj.gro

user$ java -jar SCM.jar -g proteinTraj.gro -t protein.top -o contactMapsOut
--default --multiple 1000 [--chain chainsFile]
```

B.1.4.2 Full configuration parameter list

The following will give a full list of configuration options:

```
user$ java -jar SCM.jar -help
```

B.1.4.3 Running `SCM.jar` through the webtool

On the webserver (http://smog-server.org/Shadow.html) one can build a shadow map from a SMOG formatted PDB file.

Bibliography

[1] Jeffrey K Noel and José N Onuchic. The many faces of structure-based potentials: From protein folding landscapes to structural characterization of complex biomolecules. *Computational Modeling of Biological Systems, Springer US*, pages 31–54, 2012.

[2] Paul C Whitford, Karissa Y Sanbonmatsu, and José N Onuchic. Biomolecular dynamics: order-disorder transitions and energy landscapes. *Rep. Prog. Phys.*, 75 (7):076601, 2012.

[3] J Bryngelson and P Wolynes. Intermediates and barrier crossing in a random energy model (with applications to protein folding). *J. Phys. Chem.*, 93:6902–6915, 1989.

[4] J D Bryngelson, J N Onuchic, N D Socci, and P G Wolynes. Funnels, pathways, and the energy landscape of protein folding: a synthesis. *Proteins*, 21(3):167–195, 1995.

[5] C Clementi, H Nymeyer, and J N Onuchic. Topological and energetic factors: what determines the structural details of the transition state ensemble and "en-route" intermediates for protein folding? an investigation for small globular proteins. *J. Mol. Biol.*, 298(5):937–953, 2000.

[6] Paul C Whitford, Jeffrey K Noel, Shachi Gosavi, Alexander Schug, Kevin Y Sanbonmatsu, and José N Onuchic. An all-atom structure-based potential for proteins: bridging minimal models with all-atom empirical forcefields. *Proteins*, 75(2):430–441, 2009.

[7] Sander Pronk, Szilárd Páll, Roland Schulz, Per Larsson, Pär Bjelkmar, Rossen Apostolov, Michael R Shirts, Jeremy C Smith, Peter M Kasson, David van der Spoel, Berk Hess, and Erik Lindahl. Gromacs 4.5: a high-throughput and highly parallel open source molecular simulation toolkit. *Bioinformatics*, 29(7):845–854, 2013.

[8] James C Phillips, Rosemary Braun, Wei Wang, James Gumbart, Emad Tajkhorshid, Elizabeth Villa, Christophe Chipot, Robert D Skeel, Laxmikant Kalé, and

Klaus Schulten. Scalable molecular dynamics with namd. *J. Comput. Chem.*, 26 (16):1781–1802, 2005.

[9] Jeffrey K Noel, Paul C Whitford, Karissa Y Sanbonmatsu, and José N Onuchic. Smog@ctbp: simplified deployment of structure-based models in gromacs. *Nucleic Acids Res.*, 38:W657–61, 2010.

[10] Ohad Givaty and Y Levy. Protein sliding along dna: dynamics and structural characterization. *J Mol Biol*, 385(4):1087–97, Jan 2009. doi: 10.1016/j.jmb.2008.11.016.

[11] Jeffrey K Noel, Paul C Whitford, and José N Onuchic. The shadow map: a general contact definition for capturing the dynamics of biomolecular folding and function. *J. Phys. Chem. B*, 116(29):8692–8702, 2012.

[12] Heiko Lammert, Alexander Schug, and José N Onuchic. Robustness and generalization of structure-based models for protein folding and function. *Proteins*, 77(4): 881–891, 2009.