SMOG2

User Manual

Version Beta

Center Of Theoretical Biological Physics
Rice University • Northeastern University

Authors:
Mohit Raghunathan, Jeffery Noel,
Paul Whitford
info@smog-server.org

Contents

1 Introduction **1**
- 1.1 Structure Based Model (SBM) . . . 1
- 1.2 SMOG2 . . . 2

2 Installation **3**
- 2.1 Prerequisites . . . 3
- 2.2 Configuration . . . 3
- 2.3 First Run . . . 4

3 Running Smog2 **5**
- 3.1 Run Setup . . . 5
  - 3.1.1 PDB file format . . . 5
  - 3.1.2 Preprocessing . . . 6
- 3.2 Structure Based Models . . . 6
  - 3.2.1 All-Atom Model . . . 7
  - 3.2.2 Coarse Grained Model . . . 7
- 3.3 Smog2 input options . . . 7

4 Template-Based Approach **9**
- 4.1 Introduction to templates . . . 9
- 4.2 Smog2 Templates . . . 10
  - 4.2.1 Biomolecular Information File (.bif) . . . 10
  - 4.2.2 Setting Information File (.sif) . . . 13
  - 4.2.3 Bond Information File (.b) & Nonbond Information File (.nb) . . . 16

A Errors and Warning Messages **19**

Bibliography **22**

Chapter 1

Introduction

1.1 Structure Based Model (SBM)

Although experimental studies have provided tremendous insights into the functional dynamics of biomolecular systems, computer simulations can offer the potential to bridge the static structural data with the dynamic experiments at atomic resolution. In principle, a chemistry-based forcefield that includes all the possible interactions at the atomic level should reproduce the structure and dynamics of a biomolecular system without the knowledge of experimental structural data. With the help of experimental data we can reduce the complexity of the forcefield and still access the long time scales needed to obtain the relevant dynamics. The integration of experimental data with computer simulation in a consistent manner is given by the energy landscape theory of protein folding [1].

The energy landscape theory relies on the idea that evolution has achieved folding robustness such that the interaction between the amino acid sequences present in the functionally competent states are mutually consistent. Hence the energy landscape for such sequences have an overall funnelled landscape. Computational models that take advantage of the funnelled nature of the energy landscape by imposing a native bias with the explicit inclusion of structural data in the Hamiltonian are called "Structure-based models" (SBM) [2]. Although orginally developed in the context of protein folding, SBM have been used in other facets of biomolecular systems such as stretching, oligomerization, and functional transitions.

1.2 SMOG2

SMOG2 is a complete rewrite of, SMOG@ctbp, a software which was designed to create Structure-based Models (SBM) to investigate the dynamics of proteins, RNA, and DNA [3]. SMOG2 shifts to a template-based approach to generate the strucutre-based Hamiltonian. The advantage of a template-based approach is that the user has full control of all interaction parameters (bonds, angles, dihedrals, contacts, 1-4 pairs, excluded-volume) between every atoms in the system. The flexibility enabled by this approach allows SMOG2 to be the central hub in the study of protein-protein interactions using structure-based model. An immediate advantage of SMOG2 and the template-based approach is the ability to model non-standard residues, such as modified residues and synthetic dyes, as well as to include inherently non-native interactions, such as electrostatics and non-native contacts, into the model.

In addition to the flexibility on the interaction parameters, the template based approach introduces modularity to the input data. The modularity of the input allows for easy reuse of molecular definitions and settings when developing various structure-based models.

Chapter 2

Installation

2.1 Prerequisites

Smog2 runs on all Unix-like operating systems. The prerequisites for Smog2 are

Perl Programming Language

Perl Data Language (PDL)

and the following modules, which are available through Perl module managing utility such as CPAN, or through manual installation:

`XML::Simple`

`Exporter`

`XML::Validator::Schema`

2.2 Configuration

To set the required environmental variables for smog2, run the following command in your shell.:

```
> source  /path/to/smog2/config.bash \path/to/smog2 directiory"
```

This temporarily sets the required enviromnetal variables for smog2 to run. To permanently configure smog2 on to your system without having to run the above command for each new shell session, include the following command in your shell profile file (such as the /.bashrc file) To permanently configure smog2 on to your system include the following command in your shell profile file (such as /.bashrc):

```
source  /path/to/smog2/config.bash \path/to/smog2 directiory"
```

2.3 First Run

To test your installation of smog2, run the following command:

```
> smog2
```

If configuration is successful, then you will be greeted with the following output:

```
**************************
********** SMOG2 **********
**************************
usage:  smogv2 [-o file.top] [-i input.pdb] [-g file.gro] [--help|-?]  [-t
templateFolder/] [-c extra.contacts] [-ignh]
Necessary Inputs
-i [input.pdb]:  Input pdb to generate Hamiltonian
-t [templateFolder]:  Folder containing templates of molecular and interaction
definitions
Optional Inputs
-g [file.gro]:  Output .gro file name
-o [file.top]:  Output .top file name
-s [file.shadow]:  Output .shadow file name
-n [file.ndx]:  Output .ndx file name
-c [extra.contacts]:  Input contact map file
-ignh:  Ignore hydrogens in PDB
-ignChainContacts:  Ignore contacts between chains
```

The options listed above are current options available for smog2. The next chapter will describe of the available options.

Chapter 3

Running Smog2

This chapter describes the usage of Smog2 and the structrue of the Smog2 software. It is recommended that you read this full chapter before using the software.

3.1 Run Setup

3.1.1 PDB file format

To prepare a SMOG model, a configuration must be provided in PDB format. To avoid parsing errors, please also follow these additional guidelines which preparing you PDB for the use with the software:

- Use a text editor to ensure that there are no hidden characters whichs can lead to unpredictable results.
- Only include lines that start with "ATOM" (to specify each atom), "HETATM", "TER" (to indicate a break between 2 chains) and "END" at the end of the file.
- Chain identifiers are ignored by Smog2. If you have multiple chains, insert "TER" (left justified) between chains. Smog2 will internally index the chains sequentially, starting with 1. NOTE: Do not immediately follow a TER line with an END line, without having any atoms listed.
- The file is not read past an "END" statement (ALL CAPS, left justified). If atoms appear after an END line, these atoms will be ignored by the Smog2 parser.
- Only residues, and atoms within a residue defined in the forcefield templates will be read by Smog2. Unrecognized residues, and atoms will lead to a PDB parse error, and the program will exit.

3.1.2 Preprocessing

The program has no internal knowledge of the residue structures, and uses the templates to generate the topology file. Each PDB file has to conform fully to the molecular structure defined in the template. For example, the all-atom and C_α templates provided with the program in the `$SMOG2_PATH/topology` directory treat a terminal and non-terminal residue differently. The terminal amino residues have an suffix "T" to their three-letter code (*ex.* GLY vs GLYT). This is to distinguish the slight difference in the structure, and interactions between terminal and non-terminal amino acids of the same kind as they are declared in the templates, in which the terminal amino acid has an extra oxygen atom (OXT).

A preprocessing tool (`$SMOG2_PATH/tools/adjustPDB.pl`) is provided to adjust your PDB to reflect this change. Run the preprocessing tool as follows:

```
> /SMOG_PATH/tools/adjustPDB <PDB file> <sbmMap | -f mapFile>
```

The first necessary input is the PDB file, the second necessary input is either the option `sbmMap` to use the map file provided with the program, 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
...
...
```

The "#" character is used for comments. Each line has three strings seperated by a single space. The first string is the residue name, the second string is the name to be reformatted to if that residue appears as the first residue of a chain, and the third string is the name to be reformatted to if that residue appears as the last residue of a chain. The preprocessing tool will ouput an adjusted PDB file with the suffix *.mpd*.

3.2 Structure Based Models

Smog2 supports all forms of structure-based mdels, the All-Atom [1], and the C_α [2] models are supplied by default.

[1] Whitford PC, et al. (2009) Proteins: Structure, Function, Bioinformatics 75, 430-441.
[2] Clementi C, Nymeyer H & Onuchic JN (2000) J. Mol. Biol., 298, 937-953

3.2.1 All-Atom Model

The All-Atom Hamiltonian is defined through the template files under the directory templates/allAtom/. These files define 1) The structure of an all-atom amino, nucleic acid biomolecules, and some bioinorganic atom and ligands. 2) Define the energetic settings for the All-Atom model.

To generate an All-Atom topology and Gromacs coordinates (Gro) files, simply run:
`> smog2 -i yourFile.pdb -tAA /templates/allAtom`
The following files will be generated: temp.top, temp.gro, shadow.contacts, temp.ndx, shadow.log.

3.2.2 Coarse Grained Model

Smog2 supports general coarse grained model declarations. We provide the template for the C_α model. See Chapter ?? for a description of how to define coarse variants of coarse grained models.

To generate a C_α topology, simply run:
`> smog2 -i yourFile.pdb -tCG /templates/CAlpha -contactRes /templates/allAtom`
The following files will be generated: temp.top, temp.gro, shadow.contacts, temp.ndx, shadow.log.
Note the slight difference in the input parameters for the coarse-grained model. The additional option `-contactRes` is needed for the contact map generation program to resolve all the atoms in the input PDB. Since the input PDB contains all the atoms of the residue by default, an All-Atom template is given as the contact resolution.

3.3 Smog2 input options

Smog2 expects two necessary inputs, a PDB file, and a Template folder. The table below shows all the input options currently supported by smog2.

Input Option	**Usage**	**Required/Optional**
`-i <PDB file>`	Input PDB file to build the Hamiltonian on	Required
`-tAA <Template Folder>`	Input template folder that defines the SBM Hamiltonian	Required if generating an all-atom model, otherwise the option -tCG has to be used.
`-tCG <Template Folder>`	Input template folder that defines a coarse grained SBM Hamiltonian	Required if generating a coarse grained model, otherwise the option -tAA has to be used. In addition the -contactRes flag has to be used
`-contactRes <Template Folder>`	Input template folder that defines contact resolution	Required if the -tCG option is used.
`-g <file.gro>`	Output .gro file name	Optional. Default value is temp.gro
`-o <file.top>`	Output .top file name	Optional. Default value is temp.top
`-s <file.shadow>`	Output .shadow file name	Optional. Default value is temp.shadow
`-n <file.ndx>`	Output .ndx file name	Optional. Default is temp.ndx
`-ignh`	Ignore hydrogens in PDB	Optional
`-ignChainContacts`	Ignore contacts between atoms of different chains	Optional

Chapter 4

Template-Based Approach

4.1 Introduction to templates

Smog2 offers increased versatility over the original Smog implementation[1] by shifting to a template based approach to generate the structure based Hamiltonian. The templates will allow for direct control of the structure-based Hamiltonian, including but not limited to multi-resolution models, and models the include non-specific interactions. The plug-and-generate nature of the templates has the additional advantage of forcefield portability and easy sharing of user-created variations of structure-based potentials.

Four XML formatted files form the Smog2 template framework. These four files are **absolutely necessary** when using smog2. XML format was adopted because of its consistent formating, ease of editability and readibility, 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 have to conform to. This adds an additional layer of error checking capabilities. This chapter assumes that the user knows the basics of XML formatting. Users unaware to XML formatting should refer to the W3schools' website.

The table below summarizes the template files. Subsequent sections of this chapter will illustrate through an example on how to create a template for the All-Atom structure-based model.[4].

[1] Noel JK, et al. (2010) Nucleic Acid Research 38, W657-661.

File	Definition
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

4.2 Smog2 Templates

Smog2 expects four template files in a single folder (template folder). From Chapter 1 we note that a template folder is one of the two necessary inputs to Smog2. A template folder can only contain **one** of each file type. If your template folder contains more than one file of a specific file type, the program will exit with an error.

4.2.1 Biomolecular Information File (.bif)

The Biomolecular Information File (hereon .bif) declares the structure of all the biomolecules used in your system. Smog2 uses the .bif file to parse the coordinate infromation from the PDB file. Since the PDB file only provides the coordinates of atoms, the residue definitions in the .bif file are used in conjunction with the PDB file to build the biomolecular system. We define a residue in a .bif file by declaring all the atoms in that residue, the bonds between the atoms, and the improper dihedrals between the atoms.

Residues

Residues

```
<!--ALANINE_EXAMPLE-->
                <residue name="ALAE" type="amino" atomCount=''4''>
                        <atoms>
                                <atom bType="X" nbType="Y" pairType="Z">N</atom>
                                <atom bType="X" nbType="Y" pairType="Z">CA</atom>
                                <atom bType="X" nbType="Y" pairType="Z">CB</atom>
                                <atom bType="X" nbType="Y" pairType="Z">C</atom>
                        </atoms>
                        <bonds>
        <!--BACKBONE-->
                                <bond diheGroup="bb_a">
                                        <atom>N</atom>
                                        <atom>CA</atom>
                                </bond>
                        </bonds>
                        <impropers>
                                <improper>
                                        <atom>CB</atom>
                                        <atom>CA</atom>
                                        <atom>C</atom>
                                        <atom>N</atom>
                                </improper>
                        </impropers>
                </residue>
```

Listing 4.1: Residue section of .bif file

Shown above is a declaration of a modified Alanine molecule with only four atoms, a single bond and a single improper dihedral. The attribute *name* should match the residue name used in your PDB file. The attribute residue *type* is the type of residue ALAE belongs to, in this case its a type amino residue. Finally the *atomCount* attribute allows the user to explicitly set the total number of atoms to be counted for particular residue. The global atom count is used in the energetic scaling prodecure of dihedrals and contact energies. This feature is particularly useful when using ligands in the system, as ligands are accounted as perturbation to the global Hamiltonian, and hence their atom counts are not included in the energetic scaling procedure. In this case the user would set the *atomCount* to 0. If the *atomCount* is left off, the program automatically uses the total number of atoms listed under the `<atoms>` tag.

The `<atoms>` tag encapsulates all the atoms in the biomolecule. Note that all the atoms within a specific residue in your PDB **has to be** listed here. If not, the program will terminate with an error. This is one of the reason we have the C-terminal amino residue as a unique residue seperate from the internal amino residues, as a C-terminal amino residue has an extra OXT atom. We list the atoms in the residue using the `<atom>` tag.

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 (contacts and exclusions). The pair type attribute is used in the generation of 1-4 interactions. Subsequent sections of this chapter will shed more light on these interaction type attributes.

The `<bonds>` tag contains all the bonds in your biomolecule. We list all the bonds in the residue using the `<bond>` tag. The atoms in these tag has to match the atoms declared earlier. The bond tag also has an attribute called *diheGroup* that classifies the energy group the specific bond belongs to. 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 calcuates all the angles, and dihedrals.

The `<impropers>` tag contains all the improper dihedral in the biomolecule. The tag `<improper></improper>` holds four atoms tag. The order of the four atoms in here define a specific improper dihedral within a biomolecule. This feature is used to add dihedrals that otherwise cannot be calculated dynamically by the program.

Connections

```
<connections>
<!-- AMINO/AMINO -->
                <connection name="amino-amino" resTypeA="amino" resTypeB="amino">
                        <bond diheGroup="r_a" >
                                <atom>C</atom>
                                <atom>N</atom>
                        </bond>
                <improper>
                                        <atom>O</atom>
                                        <atom>CA</atom>
                                        <atom>C</atom>
                                        <atom>N?</atom>
                </improper>
                </connection>
</connections>
```

Listing 4.2: Connection section of .bif file

In addition to defining a residue, the .bif file is also used to define how different residues are connected. Shown above is an example of how residue type amino connect with one another. The attribute *resTypeA* and *resTypeB* declares which two residue types we are connecting. The residue types used here must exist as part of your prior declaration

of biomolecules. Much of the structure of the connection element is similar to that of the residue element. We have a single bond, whereby the first atom belongs to the ith residue and the second atom belongs to the (i+1)th residue. We can also define, though not necessary, a single improper dihedral. The special character suffix "?" is used to declare atoms that belong to the (i+1)th residue. In the code listing shown above, the N atom belongs to the (i+1)th residue.

4.2.2 Setting Information File (.sif)

The Setting Information File (.sif) is used to control inter-residue dihedral ratios, contact-to-dihedral ratios, contact map settings and function declarations.

Functions

A new feature of SMOG2 is the ability to use all the available functions for a particular interaction as supported by the GROMACS molecular dynamics package. The `<functions>` tag should list all the functions that the user plans to use.

```
1 <functions>
2         <function name="sbm_bonds" directive="dihedrals" fType="1"/>
3         <function name="sbm_dihedrals" directive="dihedrals" fType="1"/>
4         <function name="sbm_contacts" directive="contacts" fType="1"/>
5 </functions>
```

Listing 4.3: Function section of .sif file

Each function is defined using the `<functions>` tag. The *name* attribute is a user-defined name for a particular function that will be used later to refer to in the .b and .nb file. The *directive* attribute is the directive under which the function *fType* falls under according GROMACS' definition of a *directive*. See the GROMACS manual, specifically table 5.5, for the definitions. For example, under the directive dihedrals, the function type 1 is a cosine dihedral function.

The only non-intutive carryover from GROMACS' definition of interaction functions, is the structure-based contact functions unique to strucutre-based models. The model utilizes the *contacts* directive to include the contact potentials between different atoms. Note that the *contacts* directive is a Smog2 only directive, it was added to differentiate between contacts and pairs. In your output topology file, the native contact interaction will be placed under the pairs directive. Smog2 uses the function type (*fType*) 1 for a 6-12 Lennard-Jones contact potential, type 2 for a 10-12 Lennard-Jones, and type 3 for Gaussian contact potential[5].

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

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

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

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.

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

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

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

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 types. The energy group ratios are contained within the `<Groups>` tag.

```
<Groups>
        <!-- NUCLEIC ENERGY GROUPS -->
        <!-- Normalized -->
                <diheGroup residueType="nucleic" name="bb_n" intraRelativeStrength="2" normalize="1"/>
                <diheGroup residueType="nucleic" name="sc_n" intraRelativeStrength="1" normalize="1"/>
        <!-- Not-Normalized -->
                <diheGroup residueType="nucleic" name="pr_n" normalize="0"/>
        <!-- Global Contacts Settings -->
                <contactGroup residueType="mixed" name="c" intraRelativeStrength="1" normalize="1"/>
        <!-- Contact/Dihedral Group ratio -->
       <groupRatios contacts="2" dihedrals="1"/>
</Groups>
```

LISTING 4.4: Energy group section of .sif file

The two classes of energy group ratios, dihedral and contact ratios, are controlled under the `<diheGroup>` 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 name of a particular energy group. The name of a particular dihedral energy group was used earlier in the .bif file when declaraing the bonds inside a particular residue. The *name* attribute used here appeared as the *diheGroup* 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 sebsequent section of this chapter.

The *normalize* attribute for each energy group is a boolean attribute (1 or 0), and is used to control if a particular energy group is to be renormalized to the total number of non-ligand atoms (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, and so the nucleic planar rigid dihedral energies will not be renormalization. But on the other hand the All-Atom model renormalizes the sidechain dihedrals, backbone dihedrals and contact energies to the total number of non-ligan atoms. As such, those energy group have the normalize option set to 1. The *intraRelativeStrength* attribute is the relative ratio of 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)

We discuss the .b and .nb template files collectively as they share similar formatting. The .b file is used to define all the bonded interactions such as 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="sbm_bonds(?,20000)">
4          <bType>X</bType>
5          <bType>X</bType>
6          </bond>
7          <!-- Alternative declaration -->
8          <bond func="sbm_bonds(?,20000)">
9          <bType>*</bType>
10         <bType>*</bType>
11         </bond>
12 </bonds>
```

Listing 4.5: Bonds section of .b file

Recall that the each atom was given a *bondType* when declared in the .bif file (line 4 in listing 4.1). Given a particular bonded interaction (bonds, angles, dihedrals, improper), the functional form for a bonded interaction is assigned based on the combination of the *bondType*s in that interaction. The first `<bond>` tag (line 3 in listing 4.5) is used assign a function called sbm_bonds() to two bonded atoms with *bondType*=X. Recall under listing 4.3, line 2, we defined the sbm_bonds() as a type 1 function under the bonds directive (harmonic bond function). The input parameters for the sbm_bonds() follows the same units and order as GROMACS expects its function to be in (see table 5.5 of GROMACS manual). For example, GROMACS expects the bond function type 1 to have arguments in the form $(\theta_0, \epsilon_{bond})$. In this case we use the special character "?" to denote the program to calculate the native angle (θ_0) from the PDB structure file. We can instead also give an empherical value for the bond angle. This fetaure is useful when added empherical terms to the Hamiltonian.

The *bType* attribute here can take either an exact bond type or a special wildcard "*" character that matches to all available *bondType*s. For the case of the All-Atom model, since all the *bondType* 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 matches the interaction to the most specific bond type description used. The user will have to be careful in declaring interactions that conflict with one another. The example shown in listing 4.5 has two redundant bond interactions defined for the All-Atom model.**

The angle interaction follow a similar form, but instead of expecting two *bType* attribute, it takes three. The *bType* attribute in this case is symmetric to the central *bType*.

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

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="sbm_dihedrals(?*3,?*0.5,3)+sbm_dihedrals(?,?,1)" diheGroup="bb_n">
2         <bType>*</bType>
3         <bType>*</bType>
4         <bType>*</bType>
5         <bType>*</bType>
6 </dihedral>
```

Listing 4.6: Dihedral section of .b file

The "sbm_dihedrals" function is classified under the dihedral directive with function type 1 in the .sif file (listing 4.3 line 3). For this case, GROMACS expects the arguments in the form $(\phi_s, \epsilon_d, mult)$. We note that one of the function has "?*3" and "?*0.5" as the input argument. As introduced earlier the special character "?" tells the program to calulate the native value from the PDB structure file. In this case we ask the program to calculate the dihedral angle for all dihedral interaction that involve the bond type combination *-*-*-*. By using the "?*3" argument, we tell the program to multiply all the ϕ_s by 3. Currently we **only support this feature for the dihedral directive**. Furthermore we can add multiple functions together when specifying an interaction: func="f(..)+g(..)+h(..)".

The special keyword "?" has limitation 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, and later contacts 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 to tell the program that the dihedral is considered under the global energy renormalization procedure. The ϵ_d will be dynamically calculated by the program.

Non-Bonded Interactions

The .nb files is used to generate non-bonded interactions such as contacts, 1-4 paris, and general Lennard-Jones interaction.

```
1 <!-- CONTACTS -->
2 <contact>
3         <nbType>*</nbType>
4         <nbType>*</nbType>
5         <func func="sbm_contacts(?,?,?)" contactGroup="c"></func>
6 </contact>
```

LISTING 4.7: Contacts section of .nb file

Listing 4.7 shows how to create a contact interaction. In addition to *bType*, we also declared a *nbType* for each atom in the .bif file (listing 4.1). A contact function also includes the additional *contactGroup* attribute that is used to classify a specific contact interaction under a specific contact energy group. We note from listing 4.4, line 9 that the contact group c was declared with *intraRelativeStrength*=1 and *normalize* option set to 1 (true).

Currently Smog2 **only** supports Lennard-Jones, and Gaussian contact functions. Since the contact interactions are unique to Smog2, the table below shows how Smog2 expects the input parameters.

Contact Type	**Function**
Lennard-Jones 6-12	f(ϵ_C,c6,c12)
Lennard-Jones 10-12	f(ϵ_C,c10,c12)
Gaussian	f(ϵ_C,ϵ_{NC},σ_G,r_0)

In code listing 4.7, the c6, and c12 are calculated automatically by the program using the PDB structure. ϵ_C is also calculated dynamically through the scaling equations.

Appendix A

Errors and Warning Messages

The table below gives a brief overview of the possible causes of error and warnings messages in SMOG2. In the current version of the program (beta.2.13.14) errors and warnings are all fatal. If Smog2 terminates without issuing one of these warnings, please email us at **info@smog-server.org**. We aim to make the warnings/errors as informative as possible, and your feedback can be very helpful in this respect.

SMOG2 errors/warnings	Causes
Fatal Error: Atom i in residue k does not have any bonds	An atom in a residue defined in the .bif does not have any bonds.
Fatal Error: Line i does not conform correctly to PDB standard.	A particular line in the PDB is not formed correctly.
Fatal Error: There was no contacts found, check shadow.log for possible errors during contact calculation step.	Either there was no contacts found during contact calculation step or there was an error generating contacts. Check the shadow.log files to see the possible cause of error as outputted by shadow.
Fatal Error: Bonds between atom i-j does not concur with bonds threshold settings.	The bond length between atom i-j is larger than the maximum value set in the .sif file
Fatal Error: Bond between atoms i-j-k is larger than the maximum allowed value set in the .sif file.	Bond between atoms i-j-k is larger than the maximum allowed value set in the .sif file.
Fatal Error: Contacts between atom i-j does not concur with contacts threshold.	The contact length between atom i-j is smaller than the minimum allowed value set in the .sif file
Fatal Error: The contact to dihedtal ratio in the .sif is less than 0.1	The contact/dihedral ratio is less than 0.1.

Fatal Error: The dihedral strength of energyGroup e cannot be less than 0	One or more of dihedral groups in the .sif has a relative strength that is negative. Check the .sif file
Fatal Error: Residue i at line j does not conform with PDB format	PDB line j does not conform with the PDB format.
Fatal Error: PDB line j does not conform with the PDB format.	The program skips all lines that do not begin with ATOM, TER, END, HETATM, BOND. Any other formatting issue will have its own form of specific error.
Fatal Error: Cannot read from "File". Program closing	Either a specific input file doesn't exist, or it is corrupt. The program failed at opening the file.
Fatal Error: PDB PARSE ERROR. Residue doesn't conform with .bif:: $< line >$	A specific residue in the PBD is ill-defined or missing atoms as declared in the .bif file.
Fatal Error: PDB PARSE ERROR. $< atom >$ doesn't exists in the .bif declaration of $< residue >$	A specific atom found in a residue in the PDB wasn't defined in the .bif declaration of that residue.
Fatal Error: PDB PARSE ERROR. Total number of atoms of $< residue >$ doens't match with .bif declaration.	A specific residue in the PBD is missing atoms as declared in the .bif file.
Fatal Error: There is a chain with only one residue. Cannot coarse grain	Since the lower limit on a coarse-grained is a single atom, a chain with only one residue doesn't have any bonded interaction. The program expects at least 4 residues per chain for coarse-graining.
Fatal Error: A specified energy group for $< residue_i >:< atom_i >$, $< residue_j >:< atom_j >$ doesn't exists	A specified energy group between some $atom_i$ and $atom_j$ doesn't exist in $residue_i$. Check the .bif file if the energy group is set correctly between the atoms. If so, check the .sif file if the specified energy is declared correctly.
Fatal Error: CONTACT FILE READ ERROR: Contact file $< fileName >$ doesn't exist, check shadow.log for possible error from Shadow Map program	There were possible errors in generating the contact map via the Shadow Map program. Check the shadow.log file for errors outputted by the Shadow Map program.
Fatal Error: CONTACT FILE READ ERROR: Cannot read additional contact file $< fileName >$. Program closing.	The addition contact file given with option -c cannot be read. The file may not exist or may be corrupted.

Fatal Error: $< atom_i >$ doesn't exists in $< residue_i >$ but a bond was defined.	A bond was defined between atoms that do not exist in a residue. Check the .bif file to make sure the correct atoms are listed.
Fatal Error: No contact function defined for nbType contacts $<$ *nbtypea* $>$-$<$ *nbtypeb* $>$	There wasn't a contact functino defined for this particular combination of nbType, check the .nb file to make sure a particular function is defined for this particular combinatino of nbType.
Fatal Error: Contact function type $< fType >$ is not defined	The current version of SMOG2 (beta.2.13.14) only supports three types of contact function. fType=1 is 6-12 Lennard-Jones, fType=2 is 10-12 Lennard-Jones, and fType=3 is Gaussian.
Fatal Error: SMOG_PATH environment variable not set	The SMOG_PATH environmental variable is not set. The install.bash script was probably not run correctly. Check to make sure the SMOG_PATH points to the directory SMOG2 resides on.
Fatal Error: Can't find Shadow. Make sure SMOG_PATH is set correctly.	The Shadow Map program used to calculate the contact map was not located. The Shadow Map program residues in the /tools subdirectory.
Fatal Error: Please specify a contact resolution when using -tCG option with -contactRes option	The Shadow Map program requires a contact resolution template to generate the contact map when generating coarse-grained topology file.
Fatal Error: INTERACTION GENERATE ERROR. There is no function for bType combination A-B/A-B-C/A-B-C-D. Check .b file	Check the .b file to make sure a functional form for this particular interaction combination is accounted for.

Bibliography

[1] J.D. Bryngelson and P.G. Wolynes. Spin-glasses and the statistical mechanics of protein folding, 1987.

[2] J.N. Onuchic and P.G. Wolynes. Theory of protein folding., 2004.

[3] Noel JK and et al. Smog@ctbp: simplified deployment of structure- based models in gromacs, 2010.

[4] 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: Structure, Function, and Bioinformatics*, 75(2):430–441, 2009. ISSN 1097-0134. doi: 10.1002/prot.22253. URL http://dx.doi.org/10.1002/prot.22253.

[5] Heiko Lammert, Alexander Schug, and José N. Onuchic. Robustness and generalization of structure-based models for protein folding and function. *Proteins: Structure, Function, and Bioinformatics*, 77(4):881–891, 2009. ISSN 1097-0134. doi: 10.1002/prot.22511. URL http://dx.doi.org/10.1002/prot.22511.