3 Hybrid Topologies for Point Mutations

This section describes the program *buildHybrid.py*, which was written to automatically prepare hybrid building blocks to be used to calculate free energy differences associated with amino acid point mutations. To this end, the program is compatible with the GROMOS 54A7 force field. Below, the mechanism of the program is explained in detail. The reader will find a quick description at the end of this section. It is a quick start guide for instant application of the program and for users, who are aware of what the program does.

3.1 Building Blocks

In GROMOS, an amino acid is represented as a text block of information. In this text block, called the *building block*, there are defined the different properties of the respective residue, such as the interaction types, the masses and the partial charges of the individual subunits. All those building blocks are gathered in the *54a7.mtb* file, which can be found in the forcefield/official/ directory in the 'GROMOS Force-Field Files' package, available on *http://gromos.net/*. The subunits are either the actual atoms or a set of combined atoms, which is called an *united atom*. For instance, if there is a CH_2 group in the structure, the group can be treated as a single united atom with a mass, that matches the sum of the mass of one C and two H atoms. This concept is also transfered to the charge and the Lennard-Jones parameters. Please note, that if the term *atom* is used, it may refer to an atom or an united atom. A similar concept is the combination of atoms to a *charge group*. Here, a few consecutive atoms are member of a charge group such that they may have different charges, however, these charges must add up to exactly 0 in a non-ionized molecule. Then, interactions reduce to dipole-dipole interactions that scale inversely to the cubed interatomic distance.

The building blocks contain a sequentially numbered list of every atom in the amino acid, henceforth just called *sequence number*, atom names, denoted as *ANM*, an assigned integer atom code for selection of the van der Waals parameters, denoted as *IACM*, a mass atom type code for selection of atomic masses, denoted as *IMCM*, an atomic charge, denoted as *CGM*, and an atomic charge group code, denoted as *ICGM*. The atoms forming a charge group must have sequential numbers. The last atom of a charge group is denoted by ICGM = 1, the others must have ICGM = 0. The number of neighbors of atom i that are excluded from non-bonded interactions with atom i is denoted as *MAE* while *MSAE* denotes atom sequence numbers j of excluded neighbors of atom i. It is required that $i < j$ and that the j's occur in ascending order. The exclusions of the last atoms of the building block are not specified here, but handled in the next building block of the molecular chain. The following shows an excerpt of the building block of isoleucine:

```
#ATOM ANM  IACM IMCM       CGM ICGM MAE MSAE
    1 N       6   14  -0.31000   0   4    2    3    4    8
    2 H      21    1   0.31000   1   1    3
    3 CA     14    3   0.00000   1   6    4    5    6    8    9   10
    4 CB     14    3   0.00000   0   4    5    6    7    8
    5 CG1    15    4   0.00000   0   2    6    7
    6 CG2    16    5   0.00000   0   0
    7 CD     16    5   0.00000   1   0
```

Not part of the excerpt are preceding atoms and the carbonyl group of the backbone. These enable to connect residues to a polypeptide but they are secondary in creating a hybrid building block. Furthermore, the excerpt is missing the specifications of all bonded interactions. We want to generate a combination of isoleucine and valine to show the procedure. Valine has the building block:

```
#ATOM  ANM   IACM  IMCM       CGM  ICGM  MAE  MSAE
    1  N        6    14   -0.31000    0    4     2     3     4     7
    2  H       21     1    0.31000    1    1     3
    3  CA      14     3    0.00000    0    6     4     5     6     7     8     9
    4  CB      14     3    0.00000    0    3     5     6     7
    5  CG1     16     5    0.00000    0    1     6
    6  CG2     16     5    0.00000    1    0
```

3.2 Creating a Hybrid Building Block

In order to calculate the free energy differences via thermodynamic integration, it is necessary to generate samples of the system with gradually inactivating compounds while activating other compounds continuously at the same time (see *PEEF*). This is known as *coupling parameter* approach. In this work, this method is used with respect to the side chain atoms of two residues in a protein. If there is defined a combination of two residues, where the side chain atoms of the first amino acid are integrated into the second one, it becomes possible to perturb the side chain with the GROMOS software (see figure 2 for an example). Up to now, the preparation of these files is a time-consuming task. Although it is possible to add *N* new atoms at any desired position and to adjust the following sequence numbers and the exclusion list with the built-in program *add_atom*, there are no additional changes done to the newly added atoms in the molecular topology building block file, denoted as *.mtb* file. In most residues, the side chain starts with sequence number 4, i.e. the CB atom. Therefore, only the atoms of the second amino acid after sequence number 4 are added to the hybrid. This holds for all supported residues (see section Supported Amino Acids) with the exception of proline, hydroxyproline, glycine and alanine. The former two have a backbone structure, that differs from all other amino acids. Alanine has no side-chain atoms beyond CB while glycine has no side chain at all. These four residues are therefore handled differently. The implemented program asks the user to type in the integer codes for two residues, then it executes add_atom as a subfunction. To this end, on a Linux distribution, there must be ensured that GROMOS++ is installed and that add_atom is globally executable. In case of isoleucine and valine, all

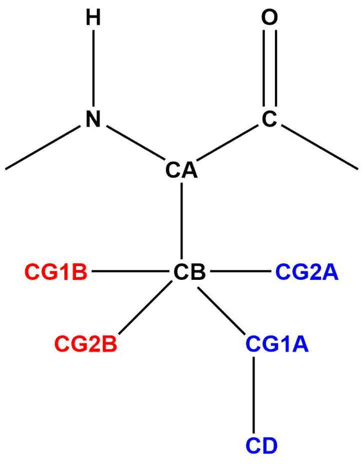

Figure 2: Illustration of the VAL_ILE hybrid building block. The side chain atoms colored in red belong to valine end state, while the ones colored in blue belong to isoleucine end state. Atoms colored in black are common to both end states.

side chain atoms of valine must be added to isoleucine. In this example, there are only CG1 and CG2 to add. The program creates a temporary building block that looks like this:

```
#ATOM ANM  IACM IMCM       CGM ICGM MAE MSAE
    1 N       6   14  -0.31000   0   4    2    3    4   10
    2 H      21    1   0.31000   1   1    3
    3 CA     14    3   0.00000   1   6    4    7    8   10   11   12
    4 CB     14    3   0.00000   0   4    7    8    9   10
    5 New     1    1   0.00000   0   0
    6 New     1    1   0.00000   0   0
    7 CG1    15    4   0.00000   0   2    8    9
    8 CG2    16    5   0.00000   0   0
    9 CD     16    5   0.00000   1   0
```

Note, that isoleucine has the longer side chain, therefore it is used as basis for the hybrid. In other cases, the side chains might be of equal length, then the aromatic character of both residues is checked and the aromatic residue is chosen as basis. If both residue side chains are of equal length and both or neither are aromatic, the first given residue is the basis of the hybrid. In hybrids consisting of proline with the 3-letter code PRO, proline is always the basis. This applies also to hydroxyproline with the 3-letter code HYP. If proline and hydroxyproline are requested, hydroxyproline is the basis. This leads to the following order, that determines which residue is the basis, starting with the first property to be checked:

$$\text{HYP} \xRightarrow{\text{no HYP}} \text{PRO} \xRightarrow{\text{no PRO}} \text{longer side chain length} \xRightarrow{\text{equal length}} \text{aromatic} \xRightarrow[\text{aromatic}]{\text{both/neither}} \text{first given.}$$

Now, there are a few tasks which have to be done and what buildHybrid.py now automatically does. These consist of inserting the actual atom names, the IACM, IMCM, CGM and ICGM values and adding the sequence numbers of the isoleucine side chain to the exclusions of the new atoms. Also, in some cases the valine atoms to add are members of the exclusion list of the valine CA and valine CB atoms and should be therefore also members of the exclusion list of the isoleucine CA and isoleucine CB atoms. To this end, the subset of exclusions of the CA and CB atoms in valine, which consists of the valine side chain atoms, gets unified with the exclusion set of the CA and CB atoms of isoleucine. To distinguish the renamed new atoms from those of the basis, the program inserts two comment lines around them with the 3-letter code for valine VAL.

As a last step, the program has to rename some of the atoms in this whole block to ensure that all atom names are unique which may be important for later analysis. To this end, in the presence of name duplicates, buildHybrid.py adds an *A* to every duplicate isoleucine atom and a *B* to every duplicate valine atom. This operations yield the following hybrid building block:

```
#ATOM ANM  IACM IMCM      CGM ICGM MAE MSAE
    1 N       6   14   -0.31000   0   4    2    3    4   10
    2 H      21    1    0.31000   1   1    3
    3 CA     14    3    0.00000   1   8    4    5    6    7    8   10
                                          11   12
    4 CB     14    3    0.00000   0   6    5    6    7    8    9   10
### VAL atoms
    5 CG1B   16    5    0.00000   0   4    6    7    8    9
    6 CG2B   16    5    0.00000   1   3    7    8    9
###
    7 CG1A   15    4    0.00000   0   2    8    9
    8 CG2A   16    5    0.00000   0   0
    9 CD     16    5    0.00000   1   0
```

Other than that, the program appends all bond types, bond angle types, dihedral angle and improper dihedral angle types to the ones of the basis residue. The required information is extracted from the valine building block on the basis of the sequence numbers of the added atoms. As illustrating example, here is the updated bond type block of the hybrid building block:

```
# bonds
#  NB
   11
#  IB   JB  MCB
    1    2    2
    1    3   21
    3    4   27
    3   10   27
    4    7   27
    4    8   27
    7    9   27
   10   11    5
   10   12   10
### VAL bonds
    4    5   27
    4    6   27
```

The last modification that is done to the .mtb file is the name. Here we have a combination of isoleucine and valine. The name inside of the building block is set to *V_I*, whereas the file name itself is set to *VAL_ILE.mtb* (see section Supported Amino Acids for letter codes).

3.3 Creating Perturbation Files

While creating the hybrid building block, the program writes two perturbation files, simultaneously. One for the direction from the basis residue to the other one and one for the opposite direction. The perturbation files consist of the side chain atoms and at this point, the program checks if the non-bonded parameters of both of the CB atoms are equal. These parameters are (i) the IACM,

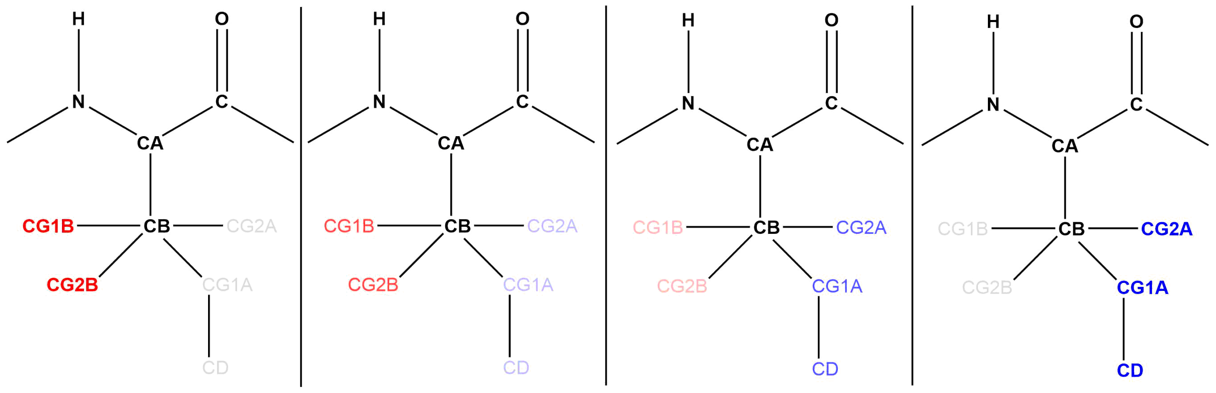

Figure 3: Illustration of the perturbation of the valine end state on the left (red) to the isoleucine end state on the right (blue). In the two end states the side chain atoms of the respective other end state are present as dummy atoms retaining all bonded interactions. The two states in the middle correspond to mixed intermediate states along a thermodynamic path morphing the side chain atoms to dummy atoms and vice versa.

(ii) the IMCM, (iii) the CGM and (iv) the ICGM. If they are not equal, the CB atom is perturbed as well. In case of isoleucine and valine, the CB atoms have equal parameters, so the generated files do not consist of the CB atom. The masses of the individual atoms are not coded with an IMCM anymore. Instead, the IMCM code for each atom is used to access the corresponding mass of the atom from the *54a7.ifp* file. Then, the actual mass, and *not* the IMCM, is written to the perturbation files. These are then stored with file type *.ptp* in the directory of execution. One of the two .ptp files that are written in our example contains the following for the direction from valine to isoleucine:

```
TITLE
template perturbation topology for perturbation of valine (state A)
to isoleucine (state B)
END
PERTATOMPARAM
# number of perturbed atoms
   5
#
# NR RES NAME IAC(A) MASS(A) CHARGE(A) IAC(B) MASS(B) CHARGE(B) ALJ  ACRF
   1   X  CG1B   16   15.035   0.00000   22    15.035   0.00000  1.0   1.0
   2   X  CG2B   16   15.035   0.00000   22    15.035   0.00000  1.0   1.0
   3   X  CG1A   22   14.027   0.00000   15    14.027   0.00000  1.0   1.0
   4   X  CG2A   22   15.035   0.00000   16    15.035   0.00000  1.0   1.0
   5   X  CD     22   15.035   0.00000   16    15.035   0.00000  1.0   1.0
```

As can be seen, the residue number *RES* is set to *X* for each atom. The RES must be set manually by the user depending on the application. Figure 3 shows the transition that is defined by the prepared .ptp file. The side chain atoms of valine gradually disappear, which corresponds to a perturbation to the *dummy atom* with IAC 22. At the same time, the side chain atoms of isoleucine, initially set to IAC 22, gradually appear.

3.4 Supported Amino Acids

The GROMOS 54A7 force field consists not only of building blocks for canonical amino acids, but also for non-canonical residues, solvent molecules and nucleotides. The implemented program supports only the canonical amino acids, some in different ionized states. Proline is supported in its common structure and with a hydroxy group, i.e. hydroxyproline. Table 1 lists all supported residues, each with its integer representation in the program and with its long and short letter code. The two latter are needed for the naming inside the files and of the files themselves. There is precise structural information in the GROMOS documentation for every one of these amino acids.

Int	Residue	Long	Short
0	Alanine	ALA	A
1	Arginine (protonated; charge +e)	ARG	R
2	Arginine (deprotonated; neutral)	ARGN	RN
3	Asparagine	ASN	N
4	Asparagine (coordinated with ZN)	ASN1	N1
5	Aspartic acid (deprotonated; charge -e)	ASP	D
6	Aspartic acid (protonated; neutral)	ASPH	DH
7	Cysteine (deprotonated; charge -0.5e)	CYS	C
8	Cysteine (protonated; neutral)	CYSH	CH
9	Cysteine (1st member of S-S bridge)	CYS1	C1
10	Cysteine (2nd member of S-S bridge)	CYS2	C2
11	Glutamine	GLN	Q
12	Glutamine acid (deprotonated; charge -e)	GLU	E
13	Glutamine acid (protonated; neutral)	GLUH	EH
14	Glycine	GLY	G
15	Histidine (protonated at ND1; neutral)	HISA	HA
16	Histidine (protonated at NE2; neutral)	HISB	HB
17	Histidine (protonated at ND1 and NE2; charge +e)	HISH	HH
18	Histidine (coupled to HEME at NE2; neutral)	HIS1	H1
19	Histidine (coupled to HEMC at NE2: neutral)	HIS2	H2
20	Hydroxyproline	HYP	PH
21	Isoleucine	ILE	I
22	Leucine	LEU	L
23	Lysine (deprotonated; neutral)	LYS	K
24	Lysine (protonated; charge +e)	LYSH	KH
25	Methionine	MET	M
26	Phenylalanine	PHE	F
27	Proline	PRO	P
28	Serine	SER	S
29	Threonine	THR	T
30	Tryptophan	TRP	W
31	Tyrosine	TYR	Y
32	Valine	VAL	V

Table 1: Supported amino acids with integer, short and long letter code.

3.5 Differences for Residues ALA, GLY, PRO and HYP

As briefly mentioned above, there are some differences if glycine, alanine, proline or hydroxyproline are requested. If proline (or hydroxyproline) is part of the hybrid residue, then a hydrogen atom will be added to the block and its sequence number is set to 2. PRO (or HYP) consists of a nitrogen atom, that normally has charge 0. It normally does not have a hydrogen atom in its structure. Because the newly added hydrogen atom should now share a charge group with the nitrogen atom, the nitrogen atom's charge must be set to the opposite charge of hydrogen. This is done by buildHybrid.py for every combination of PRO (or HYP) and a regular residue. Potentially existing side chain atoms are added to the end of the sequence of PRO (or HYP). The perturbation files are defined correspondingly, i.e. H is inactivated in the perturbation direction *to* PRO (or HYP) and activated in the opposite direction. Furthermore, the insertion of H must be considered in the exclusion list of the preceding atom with sequence number -1. Therefore, the program inserts 2 to the exclusion list of -1 and updates the number of exclusions correspondingly.

Alanine has the special property, that it has only one side chain atom, namely CB. This results in hybrid building blocks, that do not differ from the building block of the other requested residue. This is also the case for glycine, because it has no side chain atoms.

3.6 Quick Description

This section is directed to users, who want to apply the program instantly. A precise explanation is given in the preceding sections and a reading of those is recommended before using buildHybrid.py.

Prerequisites on Linux:

- Installed GROMOS++ pre-MD and analysis software.
- The program add_atom, which is part of the pre-MD and analysis software, must be globally executable. It must be added to the $PATH variable, if it has not been already during installation.
- GROMOS force-field files available on the hard drive.

Using the program. We are generating a hybrid .mtb file out of the two residues isoleucine and valine:

1. Open buildHybrid.py with a text editor.
2. Replace the file paths for the variables forceFileName and massFileName with the (absolute) file paths of the files 54a7.mtb and 54a7.ifp on your hard drive and save the file.
3. Execute python buildHybrid.py.
4. Identify the integer representing the residues in the list.
5. For isoleucine it is 21, for valine it is 32.
6. Type in the first number and hit the enter key. Then, type in the second number and hit the enter key. The order does not matter.

Output. This creates three files:

- One .mtb file with a building block based on isoleucine with newly added atoms of valine's side chain with all necessary exclusion and interaction values.
- Two .ptp files for a perturbation both from valine to isoleucine and from isoleucine to valine.

It is possible to execute the program with optional arguments, which might be convenient in certain situations. Available parameters are:

- Two integers corresponding to the respective amino acids. These must always be at the very end of the argument list and cause the program to produce the output without waiting for your input.
- -v flag (verbose) for keeping all files, that are created, processed and deleted otherwise, such as .arg files for the add_atom command and other.
- -s flag (silent) for not prompting the residue list.