Chapter 5

Topologies

5.1 Introduction

GROMACS must know on which atoms and combinations of atoms the various contributions to the potential functions (see chapter 4) must act. It must also know what parameters must be applied to the various functions. All this is described in the *topology* file `*.top`, which lists the *constant attributes* of each atom. There are many more atom types than elements, but only atom types present in biological systems are parameterized in the force field, plus some metals, ions and silicon. The bonded and special interactions are determined by fixed lists that are included in the topology file. Certain non-bonded interactions must be excluded (first and second neighbors), as these are already treated in bonded interactions. In addition, there are *dynamic attributes* of atoms - their positions, velocities and forces. These do not strictly belong to the molecular topology, and are stored in the coordinate file `*.gro` (positions and velocities), or trajectory file `*.trr` (positions, velocities, forces).

This chapter describes the setup of the topology file, the `*.top` file and the database files: what the parameters stand for and how/where to change them if needed. First, all file formats are explained. Section 5.8.1 describes the organization of the files in each force field.

Note: if you construct your own topologies, we encourage you to upload them to our topology archive at www.gromacs.org! Just imagine how thankful you'd have been if your topology had been available there before you started. The same goes for new force fields or modified versions of the standard force fields - contribute them to the force field archive!

5.2 Particle type

In GROMACS, there are three types of particles, see Table 5.1. Only regular atoms and virtual interaction sites are used in GROMACS; shells are necessary for polarizable models like the Shell-Water models [43].

Particle	Symbol
atoms	A
shells	S
virtual interaction sites	V (or D)

Table 5.1: Particle types in GROMACS

5.2.1 Atom types

Each force field defines a set of atom types, which have a characteristic name or number, and mass (in a.m.u.). These listings are found in the `atomtypes.atp` file (.atp = **a**tom **t**ype **p**arameter file). Therefore, it is in this file that you can begin to change and/or add an atom type. A sample from the `gromos43a1.ff` force field is listed below.

```
  O  15.99940 ;     carbonyl oxygen (C=O)
 OM  15.99940 ;     carboxyl oxygen (CO-)
 OA  15.99940 ;     hydroxyl, sugar or ester oxygen
 OW  15.99940 ;     water oxygen
  N  14.00670 ;     peptide nitrogen (N or NH)
 NT  14.00670 ;     terminal nitrogen (NH2)
 NL  14.00670 ;     terminal nitrogen (NH3)
 NR  14.00670 ;     aromatic nitrogen
 NZ  14.00670 ;     Arg NH (NH2)
 NE  14.00670 ;     Arg NE (NH)
  C  12.01100 ;     bare carbon
CH1  13.01900 ;     aliphatic or sugar CH-group
CH2  14.02700 ;     aliphatic or sugar CH2-group
CH3  15.03500 ;     aliphatic CH3-group
```

Note: GROMACS makes use of the atom types as a name, *not* as a number (as *e.g.* in GROMOS).

5.2.2 Virtual sites

Some force fields use virtual interaction sites (interaction sites that are constructed from other particle positions) on which certain interactions are located (*e.g.* on benzene rings, to reproduce the correct quadrupole). This is described in sec. 4.7.

To make virtual sites in your system, you should include a section `[ virtual_sites? ]` (for backward compatibility the old name `[ dummies? ]` can also be used) in your topology file, where the '`?`' stands for the number constructing particles for the virtual site. This will be '`2`' for type 2, '`3`' for types 3, 3fd, 3fad and 3out and '`4`' for type 4fdn. The last of these replace an older 4fd type (with the 'type' value 1) that could occasionally be unstable; while it is still supported internally in the code, the old 4fd type should not be used in new input files. The different types are explained in sec. 4.7.

Parameters for type 2 should look like this:

```
[ virtual_sites2 ]
```

```
; Site  from        funct  a
5       1     2     1      0.7439756
```

for type 3 like this:

```
[ virtual_sites3 ]
; Site  from              funct   a          b
5       1     2     3     1       0.7439756  0.128012
```

for type 3fd like this:

```
[ virtual_sites3 ]
; Site  from              funct   a          d
5       1     2     3     2       0.5        -0.105
```

for type 3fad like this:

```
[ virtual_sites3 ]
; Site  from              funct   theta      d
5       1     2     3     3       120        0.5
```

for type 3out like this:

```
[ virtual_sites3 ]
; Site  from              funct   a          b          c
5       1     2     3     4       -0.4       -0.4       6.9281
```

for type 4fdn like this:

```
[ virtual_sites4 ]
; Site  from                    funct   a          b          c
5       1     2     3     4     2       1.0        0.9       0.105
```

This will result in the construction of a virtual site, number 5 (first column ‘Site’), based on the positions of the atoms whose indices are 1 and 2 or 1, 2 and 3 or 1, 2, 3 and 4 (next two, three or four columns ‘from’) following the rules determined by the function number (next column ‘funct’) with the parameters specified (last one, two or three columns ‘a b . .’). Obviously, the atom numbers (including virtual site number) depend on the molecule. It may be instructive to study the topologies for TIP4P or TIP5P water models that are included with the GROMACS distribution.

Note that if any constant bonded interactions are defined between virtual sites and/or normal atoms, they will be removed by grompp (unless the option tt -normvsbds is used). This removal of bonded interactions is done after generating exclusions, as the generation of exclusions is based on “chemically” bonded interactions.

Virtual sites can be constructed in a more generic way using basic geometric parameters. The directive that can be used is [virtual_sitesn]. Required parameters are listed in Table 5.5. An example entry for defining a virtual site at the center of geometry of a given set of atoms might be:

```
[ virtual_sitesn ]
; Site   funct    from
5        1        1     2     3     4
```

Property	Symbol	Unit
Type	-	-
Mass	m	a.m.u.
Charge	q	electron
epsilon	ϵ	kJ/mol
sigma	σ	nm

Table 5.2: Static atom type properties in GROMACS

5.3 Parameter files

5.3.1 Atoms

The *static* properties (see Table 5.2 assigned to the atom types are assigned based on data in several places. The mass is listed in `atomtypes.atp` (see 5.2.1), whereas the charge is listed in `*.rtp` (.rtp = **r**esidue **t**opology **p**arameter file, see 5.6.1). This implies that the charges are only defined in the building blocks of amino acids, nucleic acids or otherwise, as defined by the user. When generating a topology (`*.top`) using the `pdb2gmx` program, the information from these files is combined.

5.3.2 Non-bonded parameters

The non-bonded parameters consist of the van der Waals parameters V (`c6` or σ, depending on the combination rule) and W (`c12` or ϵ), as listed in the file `ffnonbonded.itp`, where `ptype` is the particle type (see Table 5.1). As with the bonded parameters, entries in `[ *type ]` directives are applied to their counterparts in the topology file. Missing parameters generate warnings, except as noted below in section 5.3.4.

```
[ atomtypes ]
;name   at.num      mass      charge   ptype          V(c6)         W(c12)
    O        8  15.99940       0.000       A   0.22617E-02   0.74158E-06
   OM        8  15.99940       0.000       A   0.22617E-02   0.74158E-06
   .....

[ nonbond_params ]
  ; i    j func       V(c6)         W(c12)
    O    O    1 0.22617E-02   0.74158E-06
    O   OA    1 0.22617E-02   0.13807E-05
    .....
```

Note that most of the included force fields also include the `at.num.` column, but this same information is implied in the OPLS-AA `bond_type` column. The interpretation of the parameters V and W depends on the combination rule that was chosen in the `[ defaults ]` section of the topology file (see 5.7.1):

$$\text{for combination rule 1 :} \qquad \begin{array}{rcccll} \mathrm{V}_{ii} & = & C_i^{(6)} & = & 4\,\epsilon_i\sigma_i^6 & [\ \mathrm{kJ\ mol^{-1}\ nm^6}\ ] \\ \mathrm{W}_{ii} & = & C_i^{(12)} & = & 4\,\epsilon_i\sigma_i^{12} & [\ \mathrm{kJ\ mol^{-1}\ nm^{12}}\ ] \end{array} \tag{5.1}$$

$$\text{for combination rules 2 and 3 :} \quad \begin{array}{rcll} \mathrm{V}_{ii} & = & \sigma_i & [\text{ nm }] \\ \mathrm{W}_{ii} & = & \epsilon_i & [\text{ kJ mol}^{-1}\text{ }] \end{array} \tag{5.2}$$

Some or all combinations for different atom types can be given in the `[ nonbond_params ]` section, again with parameters V and W as defined above. Any combination that is not given will be computed from the parameters for the corresponding atom types, according to the combination rule:

$$\text{for combination rules 1 and 3 :} \quad \begin{array}{rcl} C_{ij}^{(6)} & = & \left(C_i^{(6)}\, C_j^{(6)}\right)^{\frac{1}{2}} \\ C_{ij}^{(12)} & = & \left(C_i^{(12)}\, C_j^{(12)}\right)^{\frac{1}{2}} \end{array} \tag{5.3}$$

$$\text{for combination rule 2 :} \quad \begin{array}{rcl} \sigma_{ij} & = & \frac{1}{2}(\sigma_i + \sigma_j) \\ \epsilon_{ij} & = & \sqrt{\epsilon_i\, \epsilon_j} \end{array} \tag{5.4}$$

When σ and ϵ need to be supplied (rules 2 and 3), it would seem it is impossible to have a non-zero C^{12} combined with a zero C^6 parameter. However, providing a negative σ will do exactly that, such that C^6 is set to zero and C^{12} is calculated normally. This situation represents a special case in reading the value of σ, and nothing more.

There is only one set of combination rules for Buckingham potentials:

$$\begin{array}{rcl} A_{ij} & = & (A_{ii}\, A_{jj})^{1/2} \\ B_{ij} & = & 2/\left(\frac{1}{B_{ii}} + \frac{1}{B_{jj}}\right) \\ C_{ij} & = & (C_{ii}\, C_{jj})^{1/2} \end{array} \tag{5.5}$$

5.3.3 Bonded parameters

The bonded parameters (*i.e.* bonds, bond angles, improper and proper dihedrals) are listed in `ffbonded.itp`. The entries in this database describe, respectively, the atom types in the interactions, the type of the interaction, and the parameters associated with that interaction. These parameters are then read by `grompp` when processing a topology and applied to the relevant bonded parameters, *i.e.* `bondtypes` are applied to entries in the `[ bonds ]` directive, etc. Any bonded parameter that is missing from the relevant `[ *type ]` directive generates a fatal error. The types of interactions are listed in Table 5.5. Example excerpts from such files follow:

```
[ bondtypes ]
  ; i    j  func       b0          kb
    C    O     1   0.12300     502080.
    C   OM     1   0.12500     418400.
    ......

[ angletypes ]
  ; i    j    k  func       th0         cth
   HO   OA    C     1   109.500     397.480
   HO   OA  CH1     1   109.500     397.480
   ......

[ dihedraltypes ]
```

```
  ; i    l func        q0          cq
 NR5*  NR5    2     0.000     167.360
 NR5* NR5*    2     0.000     167.360
 ......

[ dihedraltypes ]
  ; j    k func      phi0          cp   mult
    C   OA    1   180.000      16.736      2
    C    N    1   180.000      33.472      2
    ......

[ dihedraltypes ]
;
; Ryckaert-Bellemans Dihedrals
;
; aj    ak      funct
CP2     CP2     3       9.2789  12.156  -13.120 -3.0597 26.240  -31.495
```

In the `ffbonded.itp` file, you can add bonded parameters. If you want to include parameters for new atom types, make sure you define them in `atomtypes.atp` as well.

5.3.4 Intramolecular pair interactions

Extra Lennard-Jones and electrostatic interactions between pairs of atoms in a molecule can be added in the `[ pairs ]` section of a molecule definition. The parameters for these interactions can be set independently from the non-bonded interaction parameters. In the GROMOS force fields, pairs are only used to modify the 1-4 interactions (interactions of atoms separated by three bonds). In these force fields the 1-4 interactions are excluded from the non-bonded interactions (see sec. 5.4).

```
[ pairtypes ]
  ; i    j func         cs6          cs12 ; THESE ARE 1-4 INTERACTIONS
    O    O    1 0.22617E-02   0.74158E-06
    O   OM    1 0.22617E-02   0.74158E-06
    .....
```

The pair interaction parameters for the atom types in `ffnonbonded.itp` are listed in the `[ pairtypes ]` section. The GROMOS force fields list all these interaction parameters explicitly, but this section might be empty for force fields like OPLS that calculate the 1-4 interactions by uniformly scaling the parameters. Pair parameters that are not present in the `[ pairtypes ]` section are only generated when `gen-pairs` is set to "yes" in the `[ defaults ]` directive of `forcefield.itp` (see 5.7.1). When `gen-pairs` is set to "no," `grompp` will give a warning for each pair type for which no parameters are given.

The normal pair interactions, intended for 1-4 interactions, have function type 1. Function type 2 and the `[ pairs_nb ]` are intended for free-energy simulations. When determining hydration free energies, the solute needs to be decoupled from the solvent. This can be done by adding a

B-state topology (see sec. 3.12) that uses zero for all solute non-bonded parameters, *i.e.* charges and LJ parameters. However, the free energy difference between the A and B states is not the total hydration free energy. One has to add the free energy for reintroducing the internal Coulomb and LJ interactions in the solute when in vacuum. This second step can be combined with the first step when the Coulomb and LJ interactions within the solute are not modified. For this purpose, there is a pairs function type 2, which is identical to function type 1, except that the B-state parameters are always identical to the A-state parameters. For searching the parameters in the `[ pairtypes ]` section, no distinction is made between function type 1 and 2. The pairs section `[ pairs_nb ]` is intended to replace the non-bonded interaction. It uses the unscaled charges and the non-bonded LJ parameters; it also only uses the A-state parameters. **Note** that one should add exclusions for all atom pairs listed in `[ pairs_nb ]`, otherwise such pairs will also end up in the normal neighbor lists.

Alternatively, this same behavior can be achieved without ever touching the topology, by using the `couple-moltype`, `couple-lambda0`, `couple-lambda1`, and `couple-intramol` keywords. See sections sec. 3.12 and sec. 6.1 for more information.

All three pair types always use plain Coulomb interactions, even when Reaction-field, PME, Ewald or shifted Coulomb interactions are selected for the non-bonded interactions. Energies for types 1 and 2 are written to the energy and log file in separate "LJ-14" and "Coulomb-14" entries per energy group pair. Energies for `[ pairs_nb ]` are added to the "LJ-(SR)" and "Coulomb-(SR)" terms.

5.3.5 Implicit solvation parameters

Starting with GROMACS 4.5, implicit solvent is supported. A section in the topology has been introduced to list those parameters:

```
[ implicit_genborn_params ]
; Atomtype  sar     st   pi      gbr      hct
NH1         0.155   1    1.028   0.17063  0.79 ; N
N           0.155   1    1       0.155    0.79 ; Proline backbone N
H           0.1     1    1       0.115    0.85 ; H
CT1         0.180   1    1.276   0.190    0.72 ; C
```

In this example the atom type is listed first, followed by five numbers, and a comment (following a semicolon).

Values in columns 1-3 are not currently used. They pertain to more elaborate surface area algorithms, the one from Qiu *et al.* [70] in particular. Column 4 contains the atomic van der Waals radii, which are used in computing the Born radii. The dielectric offset is specified in the `*.mdp` file, and gets subtracted from the input van der Waals radii for the different Born radii methods, as described by Onufriev *et al.* [72]. Column 5 is the scale factor for the HCT and OBC models. The values are taken from the Tinker implementation of the HCT pairwise scaling method [71]. This method has been modified such that the scaling factors have been adjusted to minimize differences between analytical surface areas and GB using the HCT algorithm. The scaling is further modified in that it is not applied pairwise as proposed by Hawkins *et al.* [71], but on a per-atom (rather than a per-pair) basis.

5.4 Exclusions

The exclusions for non-bonded interactions are generated by `grompp` for neighboring atoms up to a certain number of bonds away, as defined in the `[ moleculetype ]` section in the topology file (see 5.7.1). Particles are considered bonded when they are connected by "chemical" bonds (`[ bonds ]` types 1 to 5, 7 or 8) or constraints (`[ constraints ]` type 1). Type 5 `[ bonds ]` can be used to create a connection between two atoms without creating an interaction. There is a harmonic interaction (`[ bonds ]` type 6) that does not connect the atoms by a chemical bond. There is also a second constraint type (`[ constraints ]` type 2) that fixes the distance, but does not connect the atoms by a chemical bond. For a complete list of all these interactions, see Table 5.5.

Extra exclusions within a molecule can be added manually in a `[ exclusions ]` section. Each line should start with one atom index, followed by one or more atom indices. All non-bonded interactions between the first atom and the other atoms will be excluded.

When all non-bonded interactions within or between groups of atoms need to be excluded, is it more convenient and much more efficient to use energy monitor group exclusions (see sec. 3.3).

5.5 Constraint algorithms

Constraints are defined in the `[ constraints ]` section. The format is two atom numbers followed by the function type, which can be 1 or 2, and the constraint distance. The only difference between the two types is that type 1 is used for generating exclusions and type 2 is not (see sec. 5.4). The distances are constrained using the LINCS or the SHAKE algorithm, which can be selected in the `*.mdp` file. Both types of constraints can be perturbed in free-energy calculations by adding a second constraint distance (see 5.7.5). Several types of bonds and angles (see Table 5.5) can be converted automatically to constraints by `grompp`. There are several options for this in the `*.mdp` file.

We have also implemented the SETTLE algorithm [45], which is an analytical solution of SHAKE, specifically for water. SETTLE can be selected in the topology file. See, for instance, the SPC molecule definition:

```
[ moleculetype ]
; molname       nrexcl
SOL             1

[ atoms ]
; nr    at type res nr  ren nm  at nm   cg nr   charge
1       OW      1       SOL     OW1     1       -0.82
2       HW      1       SOL     HW2     1        0.41
3       HW      1       SOL     HW3     1        0.41

[ settles ]
; OW    funct   doh     dhh
1       1       0.1     0.16333
```

```
[ exclusions ]
1       2       3
2       1       3
3       1       2
```

The `[ settles ]` directive defines the first atom of the water molecule. The settle funct is always 1, and the distance between O-H and H-H distances must be given. **Note** that the algorithm can also be used for TIP3P and TIP4P [125]. TIP3P just has another geometry. TIP4P has a virtual site, but since that is generated it does not need to be shaken (nor stirred).

5.6 pdb2gmx input files

The GROMACS program `pdb2gmx` generates a topology for the input coordinate file. Several formats are supported for that coordinate file, but `*.pdb` is the most commonly-used format (hence the name `pdb2gmx`). `pdb2gmx` searches for force fields in sub-directories of the GROMACS `share/top` directory and your working directory. Force fields are recognized from the file `forcefield.itp` in a directory with the extension `.ff`. The file `forcefield.doc` may be present, and if so, its first line will be used by `pdb2gmx` to present a short description to the user to help in choosing a force field. Otherwise, the user can choose a force field with the `-ff xxx` command-line argument to `pdb2gmx`, which indicates that a force field in a `xxx.ff` directory is desired. `pdb2gmx` will search first in the working directory, then in the GROMACS `share/top` directory, and use the first matching `xxx.ff` directory found.

Two general files are read by `pdb2gmx`: an atom type file (extension `.atp`, see 5.2.1) from the force-field directory, and a file called `residuetypes.dat` from either the working directory, or the GROMACS `share/top` directory. `residuetypes.dat` determines which residue names are considered protein, DNA, RNA, water, and ions.

`pdb2gmx` can read one or multiple databases with topological information for different types of molecules. A set of files belonging to one database should have the same basename, preferably telling something about the type of molecules (*e.g.* aminoacids, rna, dna). The possible files are:

- `<basename>.rtp`
- `<basename>.r2b` (optional)
- `<basename>.arn` (optional)
- `<basename>.hdb` (optional)
- `<basename>.n.tdb` (optional)
- `<basename>.c.tdb` (optional)

Only the `.rtp` file, which contains the topologies of the building blocks, is mandatory. Information from other files will only be used for building blocks that come from an `.rtp` file with the same base name. The user can add building blocks to a force field by having additional files with the same base name in their working directory. By default, only extra building blocks can be defined, but calling `pdb2gmx` with the `-rtpo` option will allow building blocks in a local file to replace the default ones in the force field.

5.6.1 Residue database

The files holding the residue databases have the extension `.rtp`. Originally this file contained building blocks (amino acids) for proteins, and is the GROMACS interpretation of the `rt37c4.dat` file of GROMOS. So the residue database file contains information (bonds, charges, charge groups, and improper dihedrals) for a frequently-used building block. It is better *not* to change this file because it is standard input for `pdb2gmx`, but if changes are needed make them in the `*.top` file (see 5.7.1), or in a `.rtp` file in the working directory as explained in sec. 5.6. Defining topologies of new small molecules is probably easier by writing an include topology file `*.itp` directly. This will be discussed in section 5.7.2. When adding a new protein residue to the database, don't forget to add the residue name to the `residuetypes.dat` file, so that `grompp`, `make_ndx` and analysis tools can recognize the residue as a protein residue (see 8.1.1).

The `.rtp` files are only used by `pdb2gmx`. As mentioned before, the only extra information this program needs from the `.rtp` database is bonds, charges of atoms, charge groups, and improper dihedrals, because the rest is read from the coordinate input file. Some proteins contain residues that are not standard, but are listed in the coordinate file. You have to construct a building block for this "strange" residue, otherwise you will not obtain a `*.top` file. This also holds for molecules in the coordinate file such as ligands, polyatomic ions, crystallization co-solvents, etc. The residue database is constructed in the following way:

```
[ bondedtypes ]  ; mandatory
; bonds  angles  dihedrals  impropers
     1       1          1          2  ; mandatory

[ GLY ]  ; mandatory

 [ atoms ]  ; mandatory
; name  type  charge  chargegroup
     N     N  -0.280     0
     H     H   0.280     0
    CA   CH2   0.000     1
     C     C   0.380     2
     O     O  -0.380     2

 [ bonds ]  ; optional
;atom1 atom2      b0      kb
     N     H
     N    CA
    CA     C
     C     O
    -C     N

 [ exclusions ]  ; optional
;atom1 atom2

 [ angles ]  ; optional
;atom1 atom2 atom3    th0    cth

 [ dihedrals ]  ; optional
```

```
;atom1 atom2 atom3 atom4   phi0     cp   mult

 [ impropers ]  ; optional
;atom1 atom2 atom3 atom4     q0     cq
     N    -C    CA     H
    -C   -CA     N    -O

[ ZN ]

 [ atoms ]
    ZN    ZN   2.000     0
```

The file is free format; the only restriction is that there can be at most one entry on a line. The first field in the file is the `[ bondedtypes ]` field, which is followed by four numbers, indicating the interaction type for bonds, angles, dihedrals, and improper dihedrals. The file contains residue entries, which consist of atoms and (optionally) bonds, angles, dihedrals, and impropers. The charge group codes denote the charge group numbers. Atoms in the same charge group should always be ordered consecutively. When using the hydrogen database with `pdb2gmx` for adding missing hydrogens (see 5.6.4), the atom names defined in the `.rtp` entry should correspond exactly to the naming convention used in the hydrogen database. The atom names in the bonded interaction can be preceded by a minus or a plus, indicating that the atom is in the preceding or following residue respectively. Explicit parameters added to bonds, angles, dihedrals, and impropers override the standard parameters in the `.itp` files. This should only be used in special cases. Instead of parameters, a string can be added for each bonded interaction. This is used in GROMOS-96 `.rtp` files. These strings are copied to the topology file and can be replaced by force-field parameters by the C-preprocessor in `grompp` using `#define` statements.

`pdb2gmx` automatically generates all angles. This means that for most force fields the `[ angles ]` field is only useful for overriding `.itp` parameters. For the GROMOS-96 force field the interaction number of all angles needs to be specified.

`pdb2gmx` automatically generates one proper dihedral for every rotatable bond, preferably on heavy atoms. When the `[ dihedrals ]` field is used, no other dihedrals will be generated for the bonds corresponding to the specified dihedrals. It is possible to put more than one dihedral function on a rotatable bond. In the case of CHARMM27 FF `pdb2gmx` can add correction maps to the dihedrals using the default `-cmap` option. Please refer to 4.10.4 for more information.

`pdb2gmx` sets the number of exclusions to 3, which means that interactions between atoms connected by at most 3 bonds are excluded. Pair interactions are generated for all pairs of atoms that are separated by 3 bonds (except pairs of hydrogens). When more interactions need to be excluded, or some pair interactions should not be generated, an `[ exclusions ]` field can be added, followed by pairs of atom names on separate lines. All non-bonded and pair interactions between these atoms will be excluded.

5.6.2 Residue to building block database

Each force field has its own naming convention for residues. Most residues have consistent naming, but some, especially those with different protonation states, can have many different names. The `.r2b` files are used to convert standard residue names to the force-field build block names. If

ARG	protonated arginine
ARGN	neutral arginine
ASP	negatively charged aspartic acid
ASPH	neutral aspartic acid
CYS	neutral cysteine
CYS2	cysteine with sulfur bound to another cysteine or a heme
GLU	negatively charged glutamic acid
GLUH	neutral glutamic acid
HISD	neutral histidine with N_δ protonated
HISE	neutral histidine with N_ϵ protonated
HISH	positive histidine with both N_δ and N_ϵ protonated
HIS1	histidine bound to a heme
LYSN	neutral lysine
LYS	protonated lysine
HEME	heme

Table 5.3: Internal GROMACS residue naming convention.

no `.r2b` is present in the force-field directory or a residue is not listed, the building block name is assumed to be identical to the residue name. The `.r2b` can contain 2 or 5 columns. The 2-column format has the residue name in the first column and the building block name in the second. The 5-column format has 3 additional columns with the building block for the residue occurring in the N-terminus, C-terminus and both termini at the same time (single residue molecule). This is useful for, for instance, the AMBER force fields. If one or more of the terminal versions are not present, a dash should be entered in the corresponding column.

There is a GROMACS naming convention for residues which is only apparent (except for the `pdb2gmx` code) through the `.r2b` file and `specbond.dat` files. This convention is only of importance when you are adding residue types to an `.rtp` file. The convention is listed in Table 5.3. For special bonds with, for instance, a heme group, the GROMACS naming convention is introduced through `specbond.dat` (see 5.6.7), which can subsequently be translated by the `.r2b` file, if required.

5.6.3 Atom renaming database

Force fields often use atom names that do not follow IUPAC or PDB convention. The `.arn` database is used to translate the atom names in the coordinate file to the force-field names. Atoms that are not listed keep their names. The file has three columns: the building block name, the old atom name, and the new atom name, respectively. The residue name supports question-mark wildcards that match a single character.

An additional general atom renaming file called `xlateat.dat` is present in the `share/top` directory, which translates common non-standard atom names in the coordinate file to IUPAC/PDB convention. Thus, when writing force-field files, you can assume standard atom names and no further atom name translation is required, except for translating from standard atom names to the force-field ones.

5.6.4 Hydrogen database

The hydrogen database is stored in `.hdb` files. It contains information for the `pdb2gmx` program on how to connect hydrogen atoms to existing atoms. In versions of the database before GROMACS 3.3, hydrogen atoms were named after the atom they are connected to: the first letter of the atom name was replaced by an 'H.' In the versions from 3.3 onwards, the H atom has to be listed explicitly, because the old behavior was protein-specific and hence could not be generalized to other molecules. If more than one hydrogen atom is connected to the same atom, a number will be added to the end of the hydrogen atom name. For example, adding two hydrogen atoms to `ND2` (in asparagine), the hydrogen atoms will be named `HD21` and `HD22`. This is important since atom naming in the `.rtp` file (see 5.6.1) must be the same. The format of the hydrogen database is as follows:

```
; res    # additions
         # H add type    H        i        j        k
ALA      1
         1       1       H        N        -C       CA
ARG      4
         1       2       H        N        CA       C
         1       1       HE       NE       CD       CZ
         2       3       HH1      NH1      CZ       NE
         2       3       HH2      NH2      CZ       NE
```

On the first line we see the residue name (ALA or ARG) and the number of kinds of hydrogen atoms that may be added to this residue by the hydrogen database. After that follows one line for each addition, on which we see:

- The number of H atoms added
- The method for adding H atoms, which can be any of:

  1 *one planar hydrogen,* e.g. *rings or peptide bond*
  One hydrogen atom (n) is generated, lying in the plane of atoms (i,j,k) on the plane bisecting angle (j-i-k) at a distance of 0.1 nm from atom i, such that the angles (n-i-j) and (n-i-k) are $> 90^o$.

  2 *one single hydrogen,* e.g. *hydroxyl*
  One hydrogen atom (n) is generated at a distance of 0.1 nm from atom i, such that angle (n-i-j)=109.5 degrees and dihedral (n-i-j-k)=trans.

  3 *two planar hydrogens,* e.g. *ethylene -C=CH_2, or amide -C(=O)NH_2*
  Two hydrogens (n1,n2) are generated at a distance of 0.1 nm from atom i, such that angle (n1-i-j)=(n2-i-j)=120 degrees and dihedral (n1-i-j-k)=cis and (n2-i-j-k)=trans, such that names are according to IUPAC standards [126].

  4 *two or three tetrahedral hydrogens,* e.g. *-CH_3*
  Three (n1,n2,n3) or two (n1,n2) hydrogens are generated at a distance of 0.1 nm from atom i, such that angle (n1-i-j)=(n2-i-j)=(n3-i-j)=109.47^o, dihedral (n1-i-j-k)=trans, (n2-i-j-k)=trans+120 and (n3-i-j-k)=trans+240^o.

5 *one tetrahedral hydrogen,* e.g. C_3CH
One hydrogen atom (n′) is generated at a distance of 0.1 nm from atom i in tetrahedral conformation such that angle (n′-i-j)=(n′-i-k)=(n′-i-l)=109.47°.

6 *two tetrahedral hydrogens,* e.g. $C\text{-}CH_2\text{-}C$
Two hydrogen atoms (n1,n2) are generated at a distance of 0.1 nm from atom i in tetrahedral conformation on the plane bisecting angle j-i-k with angle (n1-i-n2)=(n1-i-j)=(n1-i-k)=109.47°.

7 *two water hydrogens*
Two hydrogens are generated around atom i according to SPC [81] water geometry. The symmetry axis will alternate between three coordinate axes in both directions.

10 *three water "hydrogens"*
Two hydrogens are generated around atom i according to SPC [81] water geometry. The symmetry axis will alternate between three coordinate axes in both directions. In addition, an extra particle is generated on the position of the oxygen with the first letter of the name replaced by 'M'. This is for use with four-atom water models such as TIP4P [125].

11 *four water "hydrogens"*
Same as above, except that two additional particles are generated on the position of the oxygen, with names 'LP1' and 'LP2.' This is for use with five-atom water models such as TIP5P [127].

- The name of the new H atom (or its prefix, *e.g.* `HD2` for the asparagine example given earlier).
- Three or four control atoms (i,j,k,l), where the first always is the atom to which the H atoms are connected. The other two or three depend on the code selected. For water, there is only one control atom.

Some more exotic cases can be approximately constructed from the above tools, and with suitable use of energy minimization are good enough for beginning MD simulations. For example secondary amine hydrogen, nitrenyl hydrogen (C=NH) and even ethynyl hydrogen could be approximately constructed using method 2 above for hydroxyl hydrogen.

5.6.5 Termini database

The termini databases are stored in `aminoacids.n.tdb` and `aminoacids.c.tdb` for the N- and C-termini respectively. They contain information for the `pdb2gmx` program on how to connect new atoms to existing ones, which atoms should be removed or changed, and which bonded interactions should be added. Their format is as follows (from `gromos43a1.ff/aminoacids.c.tdb`):

```
[ None ]

[ COO- ]
[ replace ]
C  C  C  12.011  0.27
O   O1  OM  15.9994  -0.635
```

```
OXT O2 OM 15.9994 -0.635
[ add ]
2 8 O C CA N
OM 15.9994 -0.635
[ bonds ]
C O1 gb_5
C O2 gb_5
[ angles ]
O1 C O2 ga_37
CA C O1 ga_21
CA C O2 ga_21
[ dihedrals ]
N CA C O2 gd_20
[ impropers ]
C CA O2 O1 gi_1
```

The file is organized in blocks, each with a header specifying the name of the block. These blocks correspond to different types of termini that can be added to a molecule. In this example `[ COO- ]` is the first block, corresponding to changing the terminal carbon atom into a deprotonated carboxyl group. `[ None ]` is the second terminus type, corresponding to a terminus that leaves the molecule as it is. Block names cannot be any of the following: `replace`, `add`, `delete`, `bonds`, `angles`, `dihedrals`, `impropers`. Doing so would interfere with the parameters of the block, and would probably also be very confusing to human readers.

For each block the following options are present:

- `[ replace ]`
  Replace an existing atom by one with a different atom type, atom name, charge, and/or mass. This entry can be used to replace an atom that is present both in the input coordinates and in the `.rtp` database, but also to only rename an atom in the input coordinates such that it matches the name in the force field. In the latter case, there should also be a corresponding `[ add ]` section present that gives instructions to add the same atom, such that the position in the sequence and the bonding is known. Such an atom can be present in the input coordinates and kept, or not present and constructed by `pdb2gmx`. For each atom to be replaced on line should be entered with the following fields:
  - name of the atom to be replaced
  - new atom name (optional)
  - new atom type
  - new mass
  - new charge
- `[ add ]`
  Add new atoms. For each (group of) added atom(s), a two-line entry is necessary. The first line contains the same fields as an entry in the hydrogen database (name of the new atom, number of atoms, type of addition, control atoms, see 5.6.4), but the possible types of addition are extended by two more, specifically for C-terminal additions:

8 *two carboxyl oxygens, -COO$^-$*
Two oxygens (n1,n2) are generated according to rule 3, at a distance of 0.136 nm from atom i and an angle (n1-i-j)=(n2-i-j)=117 degrees

9 *carboxyl oxygens and hydrogen, -COOH*
Two oxygens (n1,n2) are generated according to rule 3, at distances of 0.123 nm and 0.125 nm from atom i for n1 and n2, respectively, and angles (n1-i-j)=121 and (n2-i-j)=115 degrees. One hydrogen (n$'$) is generated around n2 according to rule 2, where n-i-j and n-i-j-k should be read as n$'$-n2-i and n$'$-n2-i-j, respectively.

After this line, another line follows that specifies the details of the added atom(s), in the same way as for replacing atoms, *i.e.*:

- atom type
- mass
- charge
- charge group (optional)

Like in the hydrogen database (see 5.6.1), when more than one atom is connected to an existing one, a number will be appended to the end of the atom name. **Note** that, like in the hydrogen database, the atom name is now on the same line as the control atoms, whereas it was at the beginning of the second line prior to GROMACS version 3.3. When the charge group field is left out, the added atom will have the same charge group number as the atom that it is bonded to.

- `[ delete ]`
Delete existing atoms. One atom name per line.

- `[ bonds ]`, `[ angles ]`, `[ dihedrals ]` and `[ impropers ]`
Add additional bonded parameters. The format is identical to that used in the `*.rtp` file, see 5.6.1.

5.6.6 Virtual site database

Since we cannot rely on the positions of hydrogens in input files, we need a special input file to decide the geometries and parameters with which to add virtual site hydrogens. For more complex virtual site constructs (*e.g.* when entire aromatic side chains are made rigid) we also need information about the equilibrium bond lengths and angles for all atoms in the side chain. This information is specified in the `.vsd` file for each force field. Just as for the termini, there is one such file for each class of residues in the `.rtp` file.

The virtual site database is not really a very simple list of information. The first couple of sections specify which mass centers (typically called MCH_3/MNH_3) to use for CH_3, NH_3, and NH_2 groups. Depending on the equilibrium bond lengths and angles between the hydrogens and heavy atoms we need to apply slightly different constraint distances between these mass centers. **Note** that we do *not* have to specify the actual parameters (that is automatic), just the type of mass center to use. To accomplish this, there are three sections names `[ CH3 ]`, `[ NH3 ]`, and `[ NH2 ]`. For each of these we expect three columns. The first column is the atom type bound to the 2/3 hydrogens,

the second column is the next heavy atom type which this is bound, and the third column the type of mass center to use. As a special case, in the `[ NH2 ]` section it is also possible to specify `planar` in the second column, which will use a different construction without mass center. There are currently different opinions in some force fields whether an NH_2 group should be planar or not, but we try hard to stick to the default equilibrium parameters of the force field.

The second part of the virtual site database contains explicit equilibrium bond lengths and angles for pairs/triplets of atoms in aromatic side chains. These entries are currently read by specific routines in the virtual site generation code, so if you would like to extend it *e.g.* to nucleic acids you would also need to write new code there. These sections are named after the short amino acid names (`[ PHE ]`, `[ TYR ]`, `[ TRP ]`, `[ HID ]`, `[ HIE ]`, `[ HIP ]`), and simply contain 2 or 3 columns with atom names, followed by a number specifying the bond length (in nm) or angle (in degrees). **Note** that these are approximations of the equilibrated geometry for the entire molecule, which might not be identical to the equilibrium value for a single bond/angle if the molecule is strained.

5.6.7 Special bonds

The primary mechanism used by `pdb2gmx` to generate inter-residue bonds relies on head-to-tail linking of backbone atoms in different residues to build a macromolecule. In some cases (*e.g.* disulfide bonds, a heme group, branched polymers), it is necessary to create inter-residue bonds that do not lie on the backbone. The file `specbond.dat` takes care of this function. It is necessary that the residues belong to the same `[ moleculetype ]`. The `-merge` and `-chainsep` functions of `pdb2gmx` can be useful when managing special inter-residue bonds between different chains.

The first line of `specbond.dat` indicates the number of entries that are in the file. If you add a new entry, be sure to increment this number. The remaining lines in the file provide the specifications for creating bonds. The format of the lines is as follows:

```
resA atomA nbondsA resB atomB nbondsB length newresA newresB
```

The columns indicate:

1. `resA` The name of residue A that participates in the bond.
2. `atomA` The name of the atom in residue A that forms the bond.
3. `nbondsA` The total number of bonds `atomA` can form.
4. `resB` The name of residue B that participates in the bond.
5. `atomB` The name of the atom in residue B that forms the bond.
6. `nbondsB` The total number of bonds `atomB` can form.
7. `length` The reference length for the bond. If `atomA` and `atomB` are not within `length` $\pm$ 10% in the coordinate file supplied to `pdb2gmx`, no bond will be formed.
8. `newresA` The new name of residue A, if necessary. Some force fields use *e.g.* CYS2 for a cysteine in a disulfide or heme linkage.

9. `newresB` The new name of residue B, likewise.