Chapter 6
Minimization and Molecular Dynamics

A receptor model, which was energetically minimized, represents only one local minimum on the potential energy surface. Additionally, those minimized receptor models are based on homology models with more than 50 % difference in amino acid sequence compared to the template in most cases. Thus, receptor models should be refined by molecular dynamics (MD). Besides that, GPCRs, embedded in their natural surrounding, are not rigid, in contrast, they show a distinct flexibility. Thus, it is state of the art to analyze proteins by MD simulations (Carloni et al. 2002; Christen et al. 2008). In the early beginning of performing MD simulations of GPCRs the calculations were performed in gas phase without including the natural surrounding of the receptor. To avoid the destroy of the secondary and tertiary structure of the GPCR, position restraints were set onto the backbone of the transmembrane domains. However, this lead to wrong conformations of the amino acid sidechains, located at the surface of the receptor. To avoid such artefacts, the surrounding of the GPCR has to be included into the calculations. On the one hand, the surrounding stabilizes the conformation of the receptor. On the other hand, the correct surrounding allows the amino acid side chains on the receptor surface to achieve a correct conformation.

For enabling an adequate simulation box with the GPCR in its natural surrounding, at least four main steps, illustrated also in Fig. 6.1, have to be performed:

- ☐ Generate a complete model of the interesting GPCR
- ☐ Minimize the GPCR, position restraints should be put onto at least the backbone of the GPCR
- ☐ Put your GPCR correct into the lipid bilayer (see Chap. 5)
- ☐ Equilibrate the lipid bilayer around the GPCR, position restraints should be put onto at least the backbone of the GPCR
- ☐ If not already performed: center your system in the simulation box
- ☐ Solvate your lipid-GPCR-complex with water (see Chap. 5)
- ☐ Minimize your complete system; position restraints should be put onto at least the backbone of the GPCR
- ☐ Neutralize your simulation box to charge zero by putting an appropriate number of ions into the extra- or intracellular water

A. Strasser, H.-J. Wittmann, *Modelling of GPCRs,*
DOI 10.1007/978-94-007-4596-4_6, © Springer Science+Business Media Dordrecht 2013

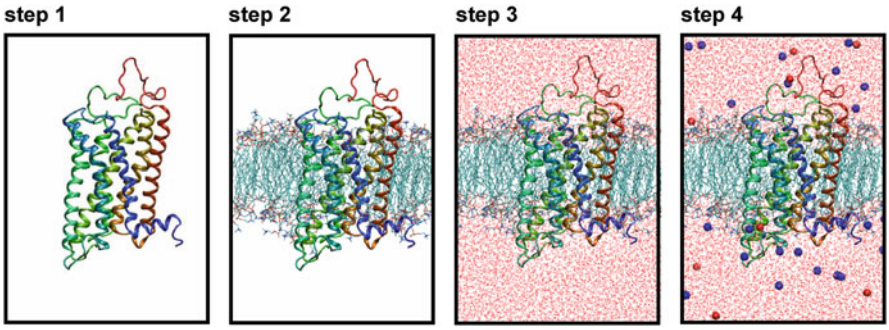

Fig. 6.1 Main steps for construction of a simulation box of a GPCR in the lipid bilayer

6.1 Generating a Complete Model of the Interesting GPCR

As already described in detail in Chap. 3, you should have designed a (homology) model of your GPCR and minimize the model in the gas phase (Fig. 6.1, step 1). In order to avoid destroying of the helical structure of the transmembrane domains, we recommend to set position constraints at least onto the backbone atoms. Therefore, you may use appropriate command of the GROMACS (http://www.gromacs.org) software package. But there is also a more flexible alternative in using LINUX-commands, as shown later on in Sect. 6.4.

6.2 Embedding the GPCR in a Lipid Bilayer

The embedding of the GPCR into a lipid bilayer (Fig. 6.1, step 2) is an important step, which has carried out very carefully. For a more detailed information see also Chap. 5.

6.3 Solvation of the Lipid-GPCR-Complex, Achiving Electroneutrality of the Simulation Box and Minimization

In the next step, the lipid-GPCR-complex should be solvated (Fig. 6.1, step 3). Some hints and pitfalls with regard to solvation of the lipid-GPCR-complex are mentioned in Chap. 5. Most modelling software allows an automatic solvation of your system. The solvation is very easy within GROMACS (http://www.gromacs.org). Here you can use the command `genbox`. If you have constructed a lipid-GPCR-complex in the file `rec_lipid.gro`, with the corresponding topology file `system.top`, you may perform the `genbox`-command for example like this:

```
> genbox -cp rec_lipid -cs -o rec_lipid_sol -p system ↵
```

The option `-cp` is used to define the file, containing the structure, that should be solvated. The option `-cs` has to be used to define the solvent. With the option `-o` you define the name of your output file. Furthermore, we recommend to use the option `-p` and give the name of the topology file, you are already using. After completion of the `genbox`-command you should visualize your solvated system (here: `rec_lipid_sol.gro`) with an appropriate software, like vmd (http://www.ks.uiuc.edu/Research/vmd/). If your system looks like the example (Fig. 6.1, step 3), all is ok and you can go on with neutralizing your system. If your ligand or protein is outward of the water shell, you have to center the actual system in the simulation box using the `editconf`-command before performing the solvation process using the file `rec_lipid.gro`, containing the lipid-GPCR-complex:

```
> editconf -f rec_lipid.gro -c -o out.gro ↵
```

Rename the file `out.gro` to `rec_lipid.gro` with the help of the `mv`-command

```
> mv out.gro rec_lipid.gro ↵
```

Now, you may again perform the `genbox`-command, as mentioned above. If the resulting simulation box looks like the one in Fig. 5.13 everything worked well, but if it looks like Fig. 5.14, the reader is referred to Sect. 5.6.

After solvation, it is recommended, to minimize the system using the commands `grompp` and `mdrun`.

```
> grompp -f mini -c rec_lipid_sol -p system ↵
```

```
> mdrun -v -s ↵
```

An example parameter file `mini.mdp`, read by `grompp` is presented below.

```
;
;       mini.mdp
;
cpp                  = /lib/cpp
;define             = -DPOSRES
constraints          = none
integrator           = steep
nsteps               = 1000
;
;       Energy minimizing
;
emtol                = 1000
emstep               = 0.01
;
pbc                  = xyz
;
nstcomm              = 1
```

```
nstlist                  = 5
rlist                    = 1.4
nstype                   = grid
coulombtype              = pme
rcoulomb                 = 1.4
epsilon_r                = 1.0
vdwtype                  = Cut-off
rvdw                     = 1.4
;DispCo                  = EnerPres
Tcoupl                   = no
Pcoupl                   = no
gen_vel                  = no
; Energy monitoring
energygrps               = system
```

Afterwards, you can start to neutralize your system (Fig. 6.1, step 4). To get information about the total charge of the system, have a look onto the output of the `grompp` command. Subsequently, you have to think about, which ions and how much you want to put into system. In general, sodium and chlorine ions are used. The concentration of sodium and chlorine ions should be chosen, that approximately physiological conditions are achieved.

Now you can neutralize your system using the command `genion`, as described in the GROMACS manual (van der Spoel et al. 2005).

After neutralization the system should be minimized again.

```
> grompp -f mini -c system -p system↵
```

```
> mdrun -v -s↵
```

If your system is minimized carefully and there are no "bugs", as described in Sect. 3.4.5, the MD simulation should work quite well.

6.4 Molecular Dynamic Simulation of your System

Now, the molecular dynamic simulation (van Gunsteren et al. 1990) can be started. In general, a MD simulation is divided into two phases: The equilibration phase and the productive phase. What does equilibration phase mean? Even if you put your GPCR very carefully in the lipid bilayer, the interactions between the lipid bilayer and the receptor are not very optimal, lets say, not equilibrated. Furthermore, during the solvation process, the water molecules are put somehow, of course in the correct density, around the lipid-GPCR-complex. But again, the interactions between the water molecules on the one hand and more importantly between the water molecules and the GPCR are not established. This means for example, no hydrogen bonds are established. If you start a molecular dynamic simulation without equilibration, the GPCR may be "destroyed", i.e. for example the helical conformation of the GPCR

is not stable. In this case, your simulation results are wrong. In the equilibration phase, the surrounding of the GPCR, the lipid bilayer and the water, should be "equilibrated" around the GPCR without modifying the structure of the GPCR. This can be done, by putting position constraints onto the GPCR. Position restraints were already introduced in context with the minimization of the system. But it has to be taken into account, that in context with molecular dynamics, distinct "equilibration" protocols should be used, in order to perform a successful and well equilibration. At the beginning of the equilibration phase, a rather high force constant k_i is to be used, but during equilibration, the force constant should be decreased gradually, until a force constant of 0 is attained (Fig. 6.2).

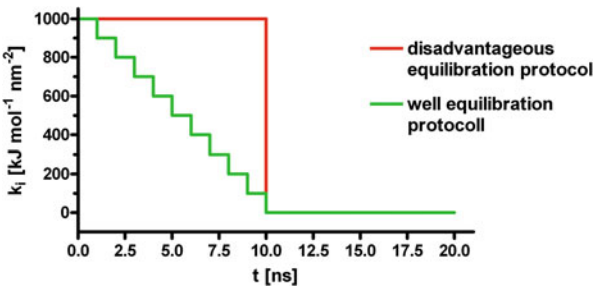

Fig. 6.2 Two different equilibration protocols for MD simulation

Of course, you can subsequently start each cycle of the equilibration protocol manually. However, it is more comfortable to establish a script `equilibrate_system`, which will be presented later on in this chapter.

First, one needs an appropriate position restraint file, which has the file-name-extension `itp` in general. Therefore one has to decide, which sites should be administered with position restraints. In the following you see a part of `gro`-file containing the coordinates of a protein in the ffG53a6-force-field notation. In the following example, the sites "`C`", "`O`", "`N`" and "`H`" should be administered with position restraints.

```
105SER      N   987    4.491    3.927    4.520
105SER      H   988    4.569    3.864    4.530
105SER     CA   989    4.495    4.049    4.604
105SER     CB   990    4.616    4.061    4.697
105SER     OG   991    4.739    4.063    4.624
105SER     HG   992    4.800    3.994    4.664
105SER      C   993    4.477    4.182    4.529
105SER      O   994    4.381    4.254    4.558
106MET      N   995    4.551    4.202    4.420
106MET      H   996    4.634    4.147    4.405
106MET     CA   997    4.533    4.321    4.333
106MET     CB   998    4.652    4.341    4.238
106MET     CG   999    4.788    4.340    4.309
106MET     SD  1000    4.803    4.453    4.452
106MET     CE  1001    4.950    4.385    4.525
```

```
  106MET       C 1002    4.399    4.329    4.255
  106MET       O 1003    4.341    4.437    4.244
  107ASP       N 1004    4.344    4.213    4.214
  107ASP       H 1005    4.396    4.128    4.216
  107ASP      CA 1006    4.210    4.204    4.149
  107ASP      CB 1007    4.189    4.061    4.099
  107ASP      CG 1008    4.087    4.049    3.985
  107ASP     OD1 1009    3.966    4.060    4.014
  107ASP     OD2 1010    4.129    4.009    3.874
  107ASP       C 1011    4.095    4.247    4.244
  107ASP       O 1012    4.024    4.344    4.216
```

A GROMACS position restraint file starts with the keyword `[position_restraints]` followed by several lines. Each line corresponds to one site and contains five columns:

First column: Number of the site (numbering according to the topology file)
Second column: function type
Third column: force constant on the x-coordinate (kJ mol^{-1} nm^{-2})
Fourth column: force constant on the y-coordinate (kJ mol^{-1} nm^{-2})
Fifth column: force constant on the z-coordinate (kJ mol^{-1} nm^{-2})

Thus, at first, the number of the sites, which should be administered with position constraints has to be determined. The `gro`-file, which should be analyzed, is named `protein.gro`, for example. The numbers of the sites, administering with position restraints, should be written into the file `site.dat`:

```
> grep " C " protein.gro | cut -c 16-21 >  site.dat↵
> grep " O " protein.gro | cut -c 16-21 >> site.dat↵
> grep " N " protein.gro | cut -c 16-21 >> site.dat↵
> grep " H " protein.gro | cut -c 16-21 >> site.dat↵
```

What does this sequence do? The command `grep " C " protein.gro` for example, looks for all lines in the file `protein.gro` which contain the string `" C "`, like shown below.

```
  105SER      C    993    4.477    4.182    4.529
  106MET      C   1002    4.399    4.329    4.255
  107ASP      C   1011    4.095    4.247    4.244
```

Note, that only lines with a blank before and after the C are printed, because, the pattern for search is `" C "`. However, you do not see this output on your screen, because the results are connected via the pipe | to the command `cut`. Why is the command `cut` used? One needs not the complete line, but only the number of the site. If you have a closer look into `protein.gro`, you see, that the site numbers are written in the columns 17–20, if the protein contains not more than 9999 sites. The option "`-c 16-21`" cuts the columns 16–21 (including a blank before and after the site number) and redirects the results in to file `site.dat`. If you would use only one >,

the file `site.dat` is created and the data are written into the new file. But be aware, if a file `site.dat` is already here in the current working directory, its data will be deleted. If the operator >> is used, all new data are appended to `site.dat`. Now, `site.dat` should contain the following information:

```
 993
1002
1011
```

After repeating the analogue commands with regard to O, N and H, the file site.dat should contain the following data:

```
 993
1002
1011
 994
1003
1012
 987
 995
1004
 998
 996
1005
```

Because the numbers are not sorted numerically, use the following command to ensure a correct order:

```
> sort -n site.dat > site_sort.dat ↵
```

To every site, a function type (second column) and a force constant for each coordinate (third to fifth column) has to be added. Therefore, we have to know, how much sites should be administered with position constraints. Because `site_sort.dat` does not contain any empty lines the appropriate number can be easily obtained using the command `wc`:

```
> wc -l site_sort.dat ↵
```

In actual example, there should be 12 lines. Thus, one has to create a new file containing "`1 1000 1000 1000`", if each force constant should have the value 1000, 12 times. This can be done using the following command:

```
> rm force.dat ↵
> set i = 1 ↵
> while ($i < = 12) ↵
>  echo "1 1000 1000 1000" >> force.dat ↵
>  @ i + + ↵
> end ↵
```

Now, both files, `site_sort.dat` and `force.dat` can be easily combined, using the command `paste`:

```
> echo "[position_restraints]" > posre_bb_1000.itp ↵
> paste site_sort.dat force.dat >> posre_bb_1000.itp ↵
```

If you performed all commands correctly, you should have the file `posre_bb_1000.itp` with the following data:

```
[position_restraints]
 987 1 1000 1000 1000
 993 1 1000 1000 1000
 994 1 1000 1000 1000
 995 1 1000 1000 1000
 996 1 1000 1000 1000
 998 1 1000 1000 1000
1002 1 1000 1000 1000
1003 1 1000 1000 1000
1004 1 1000 1000 1000
1005 1 1000 1000 1000
1011 1 1000 1000 1000
1012 1 1000 1000 1000
```

You see, that the command sequence, presented above, is very simple, in order to construct an appropriate file, containing information about position restraints. However, for your equilibration protocol, mentioned above, you will need several `itp`-files with different force constants. Therefore, the command sequence to generate the `itp`-file has to be repeated several times. Thus, it would be easier, to write an appropriate shell script.

```
 1 #!/bin/tcsh
 2
 3 set fconst = (1000 800 600 400 200 100)
 4 set nr_of_fconst = $#fconst
 5
 6 set i = 1
 7
 8 rm site.dat
 9 rm force.dat
10
11 while ($i < = $nr_of_fconst)
12
13 grep " C " protein.gro | cut -c 16-21 >> site.dat
14 grep " O " protein.gro | cut -c 16-21 >> site.dat
15 grep " N " protein.gro | cut -c 16-21 >> site.dat
16 grep " H " protein.gro | cut -c 16-21 >> site.dat
17
18 sort -n site.dat > site_sort.dat
```

```
19
20 set nr_of_res = `wc -l site_sort.dat|
   cut -d ' ' -f1`
21
22 set j = 1
23
24 while ($j < = $nr_of_res)
25  echo "1 $fconst[$i] $fconst[$i] $fconst[$i] " >>
    force.dat
26  @ j + +
27 end
28
29 echo "[position_restraints]"> posre_bb_$fconst[$i].itp
30 paste site_sort.dat force.dat >>
   posre_bb_$fconst[$i].itp
31
32 rm site.dat
33 rm force.dat
34
35 @ i + +
36
37 end
```

You may name this shell script `gen_posre`. After saving the file ensure the execute permission by using the command:

```
> chmod u + x gen_posre ↵
```

Start your shell script, by typing

```
> gen_posre ↵
```

The contents of the new `itp`-files should be proofed using an editor. With this extensive example, you should see that the linux-commands, presented in the corresponding Chap. 11 are very useful in generating and handling large files. However, the lines above only represent a rudimentary shell script which can be expanded in order to be more flexible, like checking, if a file which has to be created, is already there in the directory. Actually, the script `gen_posre` does not take care about this. However, you can use and adopt the presented shell script `gen_posre` for your own purposes.

Take into account, that the first column in the `itp`-file has to contain the site numbers of the atoms, which have to be administered with position restraints. The numbering must be according to the numbering in the topology file! You can use the `gro`-file, as we did in our example, if you have only one protein and if the protein is the first "molecule" in your `gro`-file. If this is not the case, you are suggested to adopt the script `gen_posre` with regard to the topology file. Next distinct parts of a typical GROMACS topology-file, named `protein3.top` of a protein are shown:

```
[ moleculetype ]
; Name            nrexcl
Protein_3             3

[ atoms ]
;   nr       type  resnr residue  atom   cgnr     charge       mass  typeB  chargeB  massB
     1         NL      1     ALA     N      1      0.129    14.0067   ; qtot 0.129
     2          H      1     ALA     H1     1      0.248      1.008   ; qtot 0.377
     3          H      1     ALA     H2     1      0.248      1.008   ; qtot 0.625
     4          H      1     ALA     H3     1      0.248      1.008   ; qtot 0.873
     5        CH1      1     ALA     CA     2      0.127     13.019   ; qtot 1
     6        CH3      1     ALA     CB     2          0     15.035   ; qtot 1
     7          C      1     ALA     C      3       0.45     12.011   ; qtot 1.45
     8          O      1     ALA     O      3      -0.45    15.9994   ; qtot 1
     9          N      2     PRO     N      4          0    14.0067   ; qtot 1
    10        CH1      2     PRO     CA     5          0     13.019   ; qtot 1
    11       CH2R      2     PRO     CB     5          0     14.027   ; qtot 1
    12       CH2R      2     PRO     CG     6          0     14.027   ; qtot 1
    13       CH2R      2     PRO     CD     6          0     14.027   ; qtot 1
    14          C      2     PRO     C      7       0.45     12.011   ; qtot 1.45
    15          O      2     PRO     O      7      -0.45    15.9994   ; qtot 1
    16          N      3     GLY     N      8      -0.31    14.0067   ; qtot 0.69
    17          H      3     GLY     H      8       0.31      1.008   ; qtot 1
    18        CH2      3     GLY     CA     9          0     14.027   ; qtot 1
    19          C      3     GLY     C      10      0.45     12.011   ; qtot 1.45
    20          O      3     GLY     O      10     -0.45    15.9994   ; qtot 1
    21          N      4    CYSH     N      11     -0.31    14.0067   ; qtot 0.69
    22          H      4    CYSH     H      11      0.31      1.008   ; qtot 1
    23        CH1      4    CYSH     CA     12         0     13.019   ; qtot 1
    24        CH2      4    CYSH     CB     13      0.15     14.027   ; qtot 1.15
    25          S      4    CYSH     SG     13     -0.37      32.06   ; qtot 0.78
    26          H      4    CYSH     HG     13      0.22      1.008   ; qtot 1
    27          C      4    CYSH     C      14      0.45     12.011   ; qtot 1.45
    28          O      4    CYSH     O      14     -0.45    15.9994   ; qtot 1

    29          N      5     GLY     N      15     -0.31    14.0067   ; qtot 0.69
    30          H      5     GLY     H      15      0.31      1.008   ; qtot 1
    31        CH2      5     GLY     CA     16         0     14.027   ; qtot 1
    32          C      5     GLY     C      17      0.45     12.011   ; qtot 1.45
    33          O      5     GLY     O      17     -0.45    15.9994   ; qtot 1
    34          N      6     ALA     N      18     -0.31    14.0067   ; qtot 0.69
    35          H      6     ALA     H      18      0.31      1.008   ; qtot 1
    36        CH1      6     ALA     CA     19         0     13.019   ; qtot 1
...
   439          C     52     LEU     C      193     0.45     12.011   ; qtot 6.45
   440          O     52     LEU     O      193    -0.45    15.9994   ; qtot 6
   441          N     53    HISB     N      194    -0.31    14.0067   ; qtot 5.69
   442          H     53    HISB     H      194     0.31      1.008   ; qtot 6
   443        CH1     53    HISB     CA     195        0     13.019   ; qtot 6
   444        CH2     53    HISB     CB     195        0     14.027   ; qtot 6
   445          C     53    HISB     CG     196        0     12.011   ; qtot 6
   446         NR     53    HISB     ND1    196    -0.54    14.0067   ; qtot 5.46
   447        CR1     53    HISB     CD2    196     0.14     13.019   ; qtot 5.6
   448        CR1     53    HISB     CE1    196     0.14     13.019   ; qtot 5.74
   449         NR     53    HISB     NE2    196    -0.05    14.0067   ; qtot 5.69
   450          H     53    HISB     HE2    196     0.31      1.008   ; qtot 6
   451          C     53    HISB     C      197     0.45     12.011   ; qtot 6.45
   452          O     53    HISB     O      197    -0.45    15.9994   ; qtot 6
   453          N     54     VAL     N      198    -0.31    14.0067   ; qtot 5.69
   454          H     54     VAL     H      198     0.31      1.008   ; qtot 6
   455        CH1     54     VAL     CA     199        0     13.019   ; qtot 6
   456        CH1     54     VAL     CB     199        0     13.019   ; qtot 6
   457        CH3     54     VAL     CG1    199        0     15.035   ; qtot 6
   458        CH3     54     VAL     CG2    199        0     15.035   ; qtot 6
   459          C     54     VAL     C      200     0.27     12.011   ; qtot 6.27
   460         OM     54     VAL     O1     200   -0.635    15.9994   ; qtot 5.635
   461         OM     54     VAL     O2     200   -0.635    15.9994   ; qtot 5

[ bonds ]
;  ai    aj funct            c0            c1            c2            c3
    1     2     2    gb_2
    1     3     2    gb_2
...
```

This topology file also consists of all information, which is needed for construction of a position restraint-file. The protein consists of 461 sites, which are defined from line 7–467. Thus, to extract information with regard to site number and atom, the lines 7–467 are important and they can be obtained via the command line:

```
> head -n 467 protein3.top | tail -n 461↵
```

If you perform the command, as shown above, you get the output containing 461 lines onto your xterm. However, we are not interested for the whole information of a line. Instead, if only backbone atoms should be administered with position restraints, we have to look for the corresponding site numbers (column title: `nr`) of the backbone atoms (column title: `atom`), using the following sequence of commands:

```
> head -n 467 protein3.top | tail -n 461 | tr -s ' ' |
  cut -d ' ' -f2,6 | grep ' C$' | cut -d' ' -f1 >
  site.dat↵

> head -n 467 protein3.top | tail -n 461 | tr -s ' ' |
  cut -d' ' -f2,6 | grep ' O$' | cut -d' ' -f1 >>
  site.dat↵

> head -n 467 protein3.top | tail -n 461 | tr -s ' ' |
  cut -d ' '-f2,6 | grep ' N$ ' | cut -d' ' -f1 >>
  site.dat↵

> head -n 467 protein3.top | tail -n 461 | tr -s ' ' |
  cut -d ' '-f2,6 | grep ' H$' | cut -d' ' -f1 >>
  site.dat↵
```

The output of the `head`- and `tail`-command is directed via pipe to the command `tr`. The command `tr` with the option `-s ' '` combines all subsequent white space characters to exactly one. For example

```
echo "xxx       xxx" | tr -s ' '
```
outputs: `xxx xxx`

Thus, line 7, containing information about site 1, may look like that, after using the command `tr -s` as described above:

```
  1 NL 1 ALA N 1 0.129 14.0067; qtot 0.129
```

Due to the white space character in column 1, column 2 and 6 are of interest for us: Column 2 in the line above contains information about the site and column 6 in the line above contains information about the type. Thus, the command

```
> head -n 467 protein3.top | tail -n 461 | tr -s ' ' |
cut -d' ' -f2,6↵
```

would lead to the following output (only the first seven lines are shown):

```
1 N
2 CA
3 CB
4 C
5 O
6 N
7 CA
```

Now, we have to look for all lines containing the sites, which should be administered with position restraints. In our case, this is C, O, N and H. This can be achieved by combining the command, explained above, with a corresponding `grep` command, as shown below:

```
> head -n 467 protein3.top | tail -n 461 | tr -s ' ' |
cut -d ' '-f2,6 | grep ' C$'↵
```

This command leads to the following output (only the first ten lines are shown):

```
7 C
```

Please compare the option of `grep` with the options, which were used, when dealing the same problem with the `gro`-file. In the `gro`-file, the search string could be defined as `" C "`. This means, that grep searched all lines, containing a `C` with a blank before and after the `C`. But in the actual case, one has to be aware, that there is a blank before the `C`, but there is no blank after the `C`, because, the line ends with a new line. Thus, if one searches for `"C"`, all lines with `"C"`, but also with `"CA"` and `"CB"` for example, were found. In order to avoid this, a new search criterion has to be found. This might be: Look for all lines containing a `C` at the end of a line and with a blank before the `C`. The can be achieved by `grep ' C$'`, as shown above. The `$` after the search string induces, that `grep` only searches the string at the end of a line. In order to avoid that the $ is misinterpreted as variable substitution, the single quotes have to be used instead of double quotes.

For the position restraints, only the number of the corresponding sites is of interest, thus, the long command line above has to be combined at last with the `cut`-command in the following manner:

```
> head -n 467 protein3.top | tail -n 461 | tr -s ' ' |
  cut -d' ' -f2,6 | grep ' C$' | cut -d' ' -f1↵
```

The further steps in handling the file `site.dat` are the same, as already mentioned above.

Supposing the existence of the constraint files created above, the following shell-script `equilibrate_system` can be used for equilibration of the simulation box. Be aware that the files `system.top, system.gro` (minimized simulation box, see Sect. 6.3), `md_first.mdp` (`mdp`-file for the first equilibration cycle), `md.mdp` (`mdp`-file for all following cycles) and the `itp`-files reside in the same directory as the shell-script.

```
 1  #!/bin/tcsh -f
 2
 3  set fconst = (1000 800 600 400 200 100)
 4
 5  set nr_of_fconst = $#fconst
 6
 7  set i = 1
 8
 9  while ($i < = $nr_of_fconst)
10   mkdir posre_${i}
11   cd  posre_${i}
12   cp ../system.top .
13   cp ../posre_bb_$fconst[$i].itp ./posre.itp
14
15   if ($i = = 1) then
16     cp ../system.gro .
17     cp ../md_first.mdp .
18     grompp -f md_first -o md_first -c system
       -p system
19     wait
20     mdrun -v -s md_first -e md_first -o md_first
       -c after_md -g shortlog
21     wait
22   else
23     cp ../md.mdp .
24     @ k = $i - 1
25     cp ../$posre_$k/after_md.gro ./system.gro
26     grompp -f md -o md -c system -p system
27     wait
28     mdrun -v -s md -e md -o md -c after_md
       -g shortlog
29     wait
30   endif
31   cd ..
32   @ i + +
33  end
```

The `grompp` input file `md_first.mdp` with exemplary parameters is shown below:

```
;      md_first.mdp
;      MD
;
;      Input file
;
title                    = System
cpp                      = /lib/cpp
define                   = -DPOSRES
;constraints             = all-bonds
;constraint_algorithm    = lincs
unconstrained_start      = yes
integrator               = md
tinit                    = 0
dt                       = 0.001; ps!
nsteps                   = 100000
nstcomm                  = 1
; Output control
nstxout                  = 5000
nstvout                  = 5000
nstfout                  = 0
nstlog                   = 5000
nstenergy                = 100
; Neighbor searching
nstlist                  = 10
ns_type                  = grid
pbc                      = xyz
rlist                    = 1.4
; Electrostatics and VdW
coulombtype              = PME
;rcoulomb_switch         = 0
rcoulomb                 = 1.4
epsilon_r                = 1.0
;epsilon_rf              = 7.0
vdwtype                  = Cut-off
;rvdw_switch             = 0
rvdw                     = 1.4
;DispCorr                = EnerPres
fourierspacing           = 0.12
fourier_nx               = 0
fourier_ny               = 0
fourier_nz               = 0
pme_order                = 4
ewald_rtol               = 1e-5
optimize_fft             = yes
```

```
45 ; Temperature coupling
46 tcoupl                    = berendsen
47 tc-grps                   = system
48 tau_t                     = 0.1
49 ref_t                     = 298
50 ; Energy monitoring
51 energygrps                = system
52 ; Pressure coupling is not on
53 Pcoupl                    = berendsen
54 pcoupltype                = isotropic
55 tau_p             = 0.5 0.5 0.5 0.0 0.0 0.0
56 compressibility   = 4.5e-5 4.5e-5 4.5e-5 0.0 0.0 0.0
57 ref_p             = 1.0
58 ; Generate velocites is on at 298 K.
59 gen_vel                   = yes
60 gen_temp                  = 298
61 gen_seed                  = 173529
```

In the file `md.mdp`, the parameters `unconstrained_start` and `gen_vel` should be set `no`. Afterwards, the productive simulation phase without position restraints can be started.

If the binding-mode of a ligand-receptor-complex should be analyzed via MD simulations, analogous steps, as shown above, have to be performed. Often, it is very useful, to administer the ligand with an equilibration protocol, similar to that, describe above for the receptor.

For analysis of the MD simulation, several GROMACS commands, like `g_energy`, `g_hbond`, `g_rms` and `g_traj`, for example, can be used.

It has to be taken into account, that water molecules can penetrate into the binding-pocket and mediate interactions between the ligand and receptor, as illustrated in Fig. 6.3.

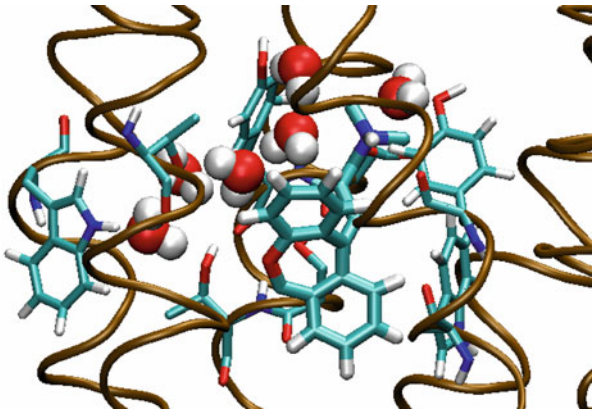

Fig. 6.3 Internal water molecules mediate the interaction between ligand and receptor