Chapter 8

Information-Driven, Ensemble Flexible Peptide Docking Using HADDOCK

Cunliang Geng*, Siddarth Narasimhan*, João P.G.L.M. Rodrigues, and Alexandre M.J.J. Bonvin

Abstract

Modeling protein-peptide interactions remains a significant challenge for docking programs due to the inherent highly flexible nature of peptides, which often adopt different conformations whether in their free or bound forms. We present here a protocol consisting of a hybrid approach, combining the most frequently found peptide conformations in complexes with representative conformations taken from molecular dynamics simulations of the free peptide. This approach intends to broaden the range of conformations sampled during docking. The resulting ensemble of conformations is used as a starting point for information-driven flexible docking with HADDOCK. We demonstrate the performance of this protocol on six cases of increasing difficulty, taken from a protein-peptide benchmark set. In each case, we use knowledge of the binding site on the receptor to drive the docking process. In the majority of cases where MD conformations are added to the starting ensemble for docking, we observe an improvement in the quality of the resulting models.

Key words Protein-peptide docking, Flexibility, Information-driven docking, Ensemble docking, HADDOCK, Molecular dynamics simulations

1 Introduction

Peptides are receiving an increasing level of attention from the wider biological and pharmaceutical communities owing to an increase in the number of peptide-based drugs and therapeutics entering the market [1, 2]. Despite their importance, there is much to be learned about the structural and dynamical properties of peptides, in particular in the context of their interactions with other biomolecules. The binding partner that peptides associate with often plays an important role in restricting/defining their

*These authors contributed equally to this work.

Electronic supplementary material: The online version of this chapter (doi:10.1007/978-1-4939-6798-8_8) contains supplementary material, which is available to authorized users.

Ora Schueler-Furman and Nir London (eds.), *Modeling Peptide-Protein Interactions: Methods and Protocols*, Methods in Molecular Biology, vol. 1561, DOI 10.1007/978-1-4939-6798-8_8, © Springer Science+Business Media LLC 2017

conformational space. Many peptides are known to exist in an intrinsically disordered state, meaning they lack a well-defined and stable folded form on the time scales that are available to experimental methods in structural biology. As a consequence, the structures of peptides are mostly known in the context of their protein receptors, which adds to the challenge of predicting in silico, the structure of these interactions.

The two most common thermodynamic models often used to describe biomolecular recognition and binding processes are induced fit [3, 4] and conformational selection [5–7]. These models were formulated based on observations from classical titration-like experiments, aimed at studying the manner in which an estimated equilibrium is achieved upon addition of a binding partner [8]. From a structural perspective, the induced fit model can be explained as a binding mechanism where the partners induce conformational changes on each other during complex formation. The conformational selection mechanism, on the other hand, predicts bound conformations are sampled naturally by the free molecules, i.e., without induction by the partner, and that partners merely select the most favorable conformation for binding (minor structural changes, such as side-chain re-orientation, may still occur upon binding).

A combination of the aforementioned mechanisms has been exploited previously to design a protocol for information-driven protein-peptide docking using HADDOCK [9]. Briefly, this protocol uses an ensemble of starting conformations for the peptide (alpha-helix, polyproline-II, and extended) that represent "ideal" conformational states of a given peptide and have been shown to feature in a large fraction of the protein-peptide interactions deposited in the Protein Data Bank (PDB) [10]. This ensemble is then docked onto the receptor structure through restraints-guided rigid-body energy minimization, and then a fraction of the models is further optimized in successive flexible refinement stages. The flexibility of the peptide is also increased compared to default HADDOCK settings. The scoring function of HADDOCK selects, at each stage, the most favored conformations, i.e., those showing the most favorable interaction with the receptor based on a set of energy criteria. This protocol thus computationally approximates a combination of the induced fit and conformational selection models.

In the protocol presented here, we suggest a way to further improve our published protein-peptide docking protocol of HADDOCK by focusing more on the fact that peptides are inherently flexible. In addition to the three most common bound conformations, we use short MD simulations of the peptides, starting from the three conformations mentioned above, to obtain more detailed information on the conformational landscape of the free peptide. Structures selected from the MD simulations that correspond to different preferred conformational states of the free

peptide supplement the three ideal structures to perform ensemble docking. Thereby, we aim at improving the conformational selection scheme in the rigid body docking stage by providing more plausible conformations of the peptide and subsequently improving the odds of success of the refinement stages. We illustrate this extended ensemble approach with cases from the benchmark of protein-peptide docking benchmark [11] and compare its performance with the standard three-conformation protocol we previously proposed [9].

2 Materials

2.1 Software Requirements

This protocol was designed and tested on a Linux cluster. Given the computational cost of molecular dynamics (MD) simulations, we recommend the use of multiple CPUs and/or GPUs. Local installation/compilation of the following programs is necessary, most of which are available for GNU/Linux and OS X operating systems:

1. *PyMOL:* PyMOL [12] is a 3D molecular structure visualization program, which can be obtained from http://pymol.org/. We use it here to generate the ideal peptide conformations (alpha-helix, polyproline-II, and extended).
2. *GROMACS*: GROMACS [13] is a molecular dynamics simulation program that includes a number of useful tools for analysis. The current protocol was run using version 5.0.4. Note that commands for versions 4.x and earlier might differ from those used here. The software is available free of charge at http://www.gromacs.org/
3. *Grace*: Grace (xmgrace) is a 2D plotting software, which provides a quick way to visualize plots generated during the execution of this protocol. It is available free of charge at http://plasma-gate.weizmann.ac.il/Grace
4. *MolProbity*: MolProbity [14] is a structure validation service that we use to assign the protonation states of Histidine residues. It can be downloaded from its GitHub repository: https://github.com/rlabduke/MolProbity. Note that other software/approaches can be used to define the charge state of Histidine residues.
5. *HADDOCK v2.2*: HADDOCK [15, 16] can be obtained free of charge for noncommercial users by filling and returning the license form available from http://www.bonvinlab.org/software/haddock2.2/download.html. Installation instructions can be found at http://www.bonvinlab.org/software/haddock2.2/installation.html. Moreover, the software can be used via a user-friendly web server [17, 18]. This protocol, however, makes use of a locally installed version of HADDOCK.

6. *Crystallography and NMR System (CNS) v1.3*: CNS [19, 20] is the engine used for energy minimization and molecular dynamics simulations in HADDOCK. Therefore, it is a main requirement for running HADDOCK. Note that HADDOCK v2.2 is designed to work with CNS v1.3, but recompiled using additional source code provided together with HADDOCK (see the cns1.3 directory in the HADDOCK distribution). The program is freely available for nonprofit users from http://cns-online.org/v1.3/
7. *NACCESS:* NACCESS is a useful tool that can be used to calculate the solvent accessible surface area of a molecule from a PDB structure file for both proteins and nucleic acids. It is free for academic users and can be obtained from http://www.bioinf.manchester.ac.uk/naccess/. A free alternative can be obtained from http://freesasa.github.io/ [21].
8. *ProFit*: ProFit is a protein least squares fitting program with many powerful features including flexible selection of fitting zones and atoms, calculation of RMS over different zones or atoms, etc. It can be obtained free of charge for academic users at http://www.bioinf.org.uk/software

2.2 Data Requirements

The structure of the receptor, preferably in the bound conformation, should be available (e.g., from the PDB, or via homology modelling) and the peptide sequence should be known. Additionally, for information-driven docking, experimental data pertaining to the interaction between the protein and peptide should be available to define the binding site on the receptor. The more information is available, the higher the chances for correct resulting models of the protein-peptide complex. Such information can be obtained from a variety of experimental techniques such as mutagenesis, chemical cross-linking, NMR chemical shift perturbations, etc. [22–24], or bioinformatics predictions (e.g., CPORT [25, 26]), all of which can be used to drive the docking in HADDOCK.

3 Methods

This protocol is divided into five major stages (summarized in Fig. 1):

1. Building the peptide in three extreme conformations.
2. Running MD simulations (50 ns) in explicit water for the peptide conformations built in **step 1**.
3. Analysis of the MD trajectories by Dihedral Principal Component Analysis (dPCA) and selection of the 30 most populated conformational states of the peptides.

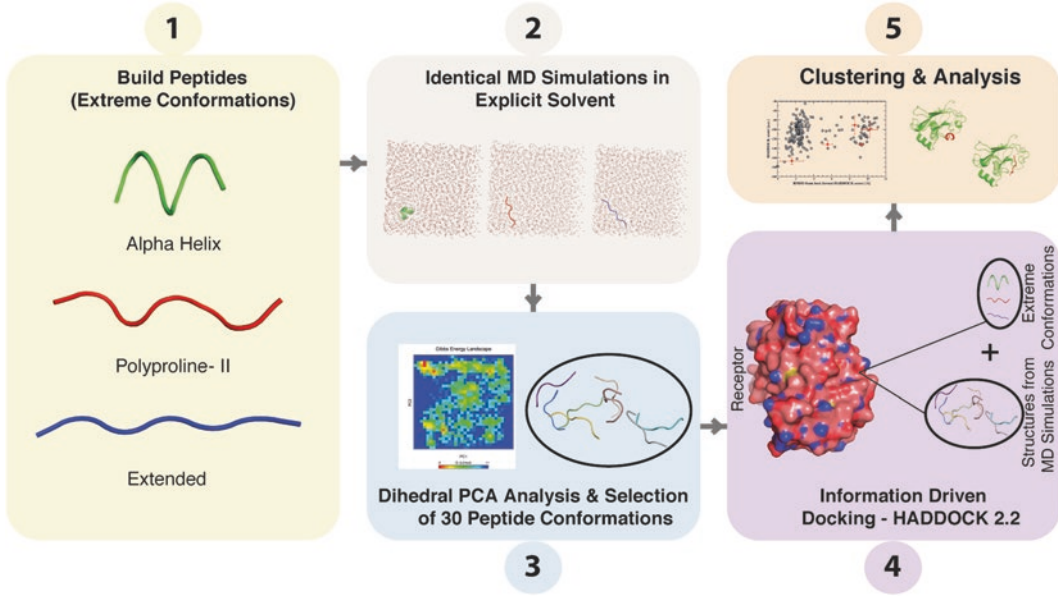

Fig. 1 Schematic overview of the workflow described in our protocol

4. Protein-peptide docking from an ensemble of 30 structures obtained from MD plus the three extreme conformations built in **step 1**, including available information on the binding sites, using HADDOCK 2.2.
5. Analysis of the docking solutions to select the best models.

To execute this protocol, the user is required to have working knowledge of a command-line interface and, preferably, experience with running MD simulations using GROMACS. In the following sections, commands are indicated in Courier font, and start with a ">" sign (note that here a command that should be given as a single line—i.e., indicated by a single ">," could span multiple lines). Text between < > in a command should be replaced by the proper selection/value.

For the sake of demonstration, unless otherwise specified, we will illustrate all the following steps using the complex of the TRAF domain of TRAF2 with the LMP1 binding peptide (PDB ID: 1CZY, *see* Table 1). All the necessary information to run this example case is provided in the supplementary material.

3.1 Generating Peptide Conformations for MD Simulations

As described in our original protein-peptide docking protocol [9] using the HADDOCK webserver, we use the *build_seq.py* script written by Robert L. Campbell to generate the starting structures of the peptides for MD simulations in PyMOL. The following steps describe this procedure, with **steps 3** and **4** describing the procedure to cap residues at the N- and C-termini. GROMACS can perform terminal capping during topology generation,

Table 1
Statistics of the six protein-peptide complexes used in the case study

Case difficulty	PDB ID complex	PDB ID free protein	Number of protein residues	Number of peptide residues	$RMSD_{bound/extended}$ (Å)
Easy ($RMSD_{bound/extended}$ ≤4 Å)	1DDV	1I2H	104	6	2.58
	1LVM	1LVB	214	6	1.54
Medium (4 Å < $RMSD_{bound/extended}$ ≤ 8 Å)	1CZY	1CZZ	168	7	1.94
	1D4T	1D1Z	101	11	3.27
Hard ($RMSD_{bound/extended}$ >8 Å)	1HC9	2ABX	74	13	11.04
	1NX1	1ALV	173	11	6.11

The classification of the case difficulty is based on Trellet et al. [9]

however, if one uses the AMBER99SB-ILDN force field [27] (which we will use in this protocol), it is necessary to manually add standard capping residues Acetyl and *N*-Methyl (abbreviated as ACE and NME). These steps can be ignored if the peptide has charged termini.

1. Start PyMOL and load the build_seq.py script from the PyMOL command line interface by typing:
   ```
   > run build_seq.py
   ```
2. Build the structure:
   ```
   > build_seq <peptide_name>, <peptide_seq>,
   ss=<secondary_structure: helix, beta, or
   polypro>
   ```
   For example, to create an alpha-helical conformation of the peptide of the case 1CZY (peptide sequence: PQQATDD), type:
   ```
   > build_seq alpha-peptide, PQQATDD, ss=helix
   ```
3. To add the capping residue to the N-terminus, first select the Nitrogen atom of the first residue (numbered as 2 by default in PyMOL) by typing the following command:
   ```
   > select pk1, name n and resi 2
   ```
   Alternatively, simply select the proper atom by clicking on it with the mouse in "editing mode," for which using a stick representation can be useful. Then select from the PyMOL menu:
   "Build > Residue > Acetyl"
4. To add the capping residue to the C-terminus, select the Carbonyl carbon atom of the last residue by typing the following command (if you have followed the previous step, ensure that you have deselected all atoms before proceeding):
   ```
   > select pk1, name c and resi <residue_number>
   ```

Alternatively, simply select the proper atom by clicking on it with the mouse. Then select from the PyMOL menu: "Build > Residue > N-Methyl"

5. Save the molecule by typing:
   ```
   > save <peptide_name>
   ```
   Or you could do it by clicking "File > Save Molecule".
   Repeat these steps to create all three starting conformations.

3.2 System Preparation for Running the MD Simulations with GROMACS 5.0.4

This protocol has been designed using the AMBER99SB-ILDN force field with periodic boundary conditions. To facilitate the combined analysis of the MD trajectories originating from different peptide conformations, it is recommended to make sure that every simulation contains the same number of water molecules. An example of a MD parameters file (*.mdp) suited for use with the AMBER99SB-ILDN force field is provided in the supplementary material. The commands described in the following subsections can also be performed by running the script *automd.sh* provided in the supplementary material (see the README section at the top of the script for instructions).

3.2.1 Determination of the Optimal Box Dimensions

We will perform the MD simulation in a rhombic dodecahedral box, to minimize the volume of the simulation cell. The dimensions of the box should be selected carefully to avoid interactions between neighboring periodic images. To determine the appropriate box dimensions, follow these steps:

1. Use the peptide in its extended conformation to determine the optimal box dimensions, considering that this represents the conformation with longest end-to-end distance.
2. Prepare the topology for the extended peptide:
   ```
   > gmx pdb2gmx -f protein.pdb -o protein.gro
   -ignh -ff amber99sb-ildn -water tip3p
   ```
3. Prepare a box that accommodates the peptide in its center and ensure that the minimum distance from the box edge to the peptide is at least half the nonbonded cutoff (as per the minimum image convention):
   ```
   > gmx editconf -f protein.pdb -o protein_
   pbc.gro -bt dodecahedron -d 1.0
   ```
4. Note the "**new box vectors**" value in the last lines of the output. This is the box vector that you must use with the "**-box**" flag during box preparation for the other peptides. An example is shown:
   ```
   system size : 2.108 2.098 0.985 (nm)
   diameter : 2.786(nm)
   center : 2.581 0.469 1.224 (nm)
   box vectors : 2.109 2.098 0.985 (nm)
   box angles : 90.00 90.00 90.00 (degrees)
   ```

```
box volume : 4.36 (nm^3)
shift : 1.009 3.120 0.468 (nm)
new center : 3.589 3.589 1.692 (nm)
new box vectors : 4.786 4.786 4.786 (nm)
new box angles : 60.00 60.00 90.00 (degrees)
new box volume : 77.51 (nm^3)
```

3.2.2 Determination of the Optimal Number of Water Molecules

Given the same box dimensions, the three peptides will accommodate a slightly different number of water molecules (to fill the box completely), depending on a variety of factors including surface areas and volumes. Therefore, it is necessary to determine this and use the smallest number of water molecules among the three systems (again to facilitate the combined analysis later—but not per se a requirement in principle). To determine this number of water molecules that the simulation system can accommodate, follow these steps:

1. Prepare the topology for the extended peptide:

   ```
   > gmx pdb2gmx -f protein.pdb -o protein.gro
   -ignh -ff amber99sb-ildn -water tip3p
   ```

2. Prepare the box, defining the box vector with the value obtained in Subheading 3.2.1:

   ```
   >gmx editconf -f protein.pdb -o protein_pbc.
   gro -bt dodecahedron -box 4.786
   ```

3. Run a first energy minimization in vacuum:

   ```
   >gmx grompp -f vacuum.mdp -c protein_pbc.
   gro -p topol.top -o protein_vac.tpr
   > gmx mdrun -v -deffnm protein_vac
   ```

4. Solvate the system:

   ```
   >gmx solvate -cp protein_vac.gro -cs spc216.
   gro -p topol.top -o protein_solvated.gro
   ```

5. Note the number of water molecules, referred to as "**SOL molecules**" that were added from the output:

   ```
   Output configuration contains 7605 atoms in
   2508 residues
   Volume : 77.518 (nm^3)
   Density : 983.104 (g/l)
   Number of SOL molecules: 2499
   ```

6. Repeat the above steps for all the peptides in all the conformations and note the lowest number of the three cases. Solvate the other two peptides (which are not the lowest) by using the "**-maxsol**" flag to set the optimal number of water molecules:

   ```
   > gmx solvate -cp protein_vac.gro -cs spc216.gro
   -p topol.top -o protein_solvated.gro -maxsol
   2499
   ```

3.2.3 System Preparation and Production MD

The following steps prepare the system for the production MD simulation. They must be repeated for all three peptide conformations. Since the initial energy minimization in vacuum and solvation has been done in the previous optimization steps, we can proceed with neutralizing the electrical charge of the system. Note that the various parameter files (.mdp) are provided in the supplementary material.

1. Add counter ions to the system to neutralize its electrical charge:
   ```
   > gmx grompp -f ions.mdp -c protein_solvated.gro -p topol.top -o protein_pp_ionize.tpr
   > gmx genion -s protein_pp_ionize.tpr -p topol.top -o protein_ionized.gro –neutral
   ```
   If and when prompted to "Select a continuous group of solvent molecules," choose the group "SOL" or "non-Protein" or "Water" as they are all the same in our system.
2. Run energy minimization:
   ```
   > gmx grompp -f ions.mdp -c protein_ionized.gro -p topol.top -o protein_neutral_relaxed.tpr
   > gmx mdrun -v -deffnm protein_neutral_relaxed
   ```
3. Run a first equilibration under NVT conditions (constant volume):
   ```
   > gmx grompp -f nvt.mdp -c protein_neutral_relaxed.gro -p topol.top -o protein_nvt.tpr
   > gmx mdrun -v -deffnm protein_nvt
   ```
4. Continue the equilibration under NPT conditions (constant pressure):
   ```
   > gmx grompp -f npt.mdp -c protein_nvt.gro -p topol.top -o protein_npt.tpr
   > gmx mdrun -v -deffnm protein_npt -nt 48 -ntmpi 12 -pin on
   ```
5. Progressively release the position restraints during successive short MD runs:
   ```
   > sed -e 's/1000  1000  1000/ 100   100   100/g' posre.itp > tmp.itp && mv tmp.itp posre.itp
   > gmx grompp -f npt.mdp -c protein_npt.gro -p topol.top -o protein_npt_progrel100.tpr
   > gmx mdrun -v -deffnm protein_npt_progrel100
   > sed -e 's/100 100 100/ 10 10 10/g' posre.itp > tmp.itp && mv tmp.itp posre.itp
   > gmx grompp -f npt.mdp -c protein_npt_progrel100.gro -p topol.top -o protein_npt_progrel10.tpr
   > gmx mdrun -v -deffnm protein_npt_progrel10
   ```

6. Run a final short unrestrained equilibration MD step:
```
> gmx grompp -f unrestrained.mdp -c protein_
npt_progrel10.gro -p topol.top -o protein_
all_set.tpr
> gmx mdrun -v -deffnm protein_all_set
```
7. Run the production MD simulation:
```
> gmx grompp -f production.mdp -c protein_
all_set.gro -p topol.top -o protein_md.tpr
> gmx mdrun -v -deffnm protein_md
```
Please bear in mind that in all the steps involving energy minimization and MD simulations (which start with **gmx mdrun**), it is likely that GROMACS might use all of the processor(s) available to the user. It is possible that **gmx mdrun** may then give error messages if the system cannot be parallelized with the given conditions. In such cases, the user must set the number of threads and the PME ranks by using the -**nt** and -**ntmpi** flags with the **gmx mdrun** command. If one wishes to optimize parallelization, GROMACS offers a utility (**gmx tune_pme**) to find the optimal PME conditions for a given number of parallel processes (see the online GROMACS manual pages for more information, http://manual.gromacs.org/archive/5.0.4/programs/gmx-tune_pme.html).

3.3 Clustering by Dihedral Principal Component Analysis (dPCA)

The protocol to perform Dihedral PCA was adapted from the GROMACS documentation (http://www.gromacs.org/Documentation/How-tos/Dihedral_PCA). It consists of the following steps:

1. Concatenate the various trajectories (three in this particular case):
```
> gmx trjcat -f <path_trajectory_1> <path_
trajectory_2> <path_trajectory_3> -o all_
traj.xtc –settime
```
Ensure that you set the proper start time for the trajectories, which in our case would be 0, 50,000, and 100,000 in picoseconds (ps). This will not be done automatically. Hence, we use the -**settime** flag.
2. Make an index file containing all residues except the capping residues:
```
> gmx make_ndx -f protein_md.tpr -o index.
ndx
```
Follow the interactive menu to isolate the indices of the peptide residues without the capping groups.
3. Remove the capping groups from the trajectory:
```
> gmx trjconv -f all_traj.xtc -s protein_
md.tpr -n index.ndx -o protein_uncapped.xtc
```

Choose the index group corresponding to the peptide residues without the capping groups that were made in the previous step.

4. Create a topology for the peptide that excludes the capping residues:

```
> gmx convert-tpr -s protein_md.tpr -n index.
ndx -o protein_uncapped.tpr
```

Select the same index group as the previous step.

5. Now create a topology containing only the backbone atoms of all residues and also a trajectory file with the corresponding coordinates:

```
> gmx convert-tpr -s protein_uncapped.tpr
-o protein_bb.tpr
> gmx trjconv -f protein_uncapped.xtc -s
protein_uncapped -o protein_bb.xtc
```

When prompted, in both cases, choose the index group titled "Backbone," which by default would be option 4.

6. Create an index group for the backbone dihedral angles by creating an angle index file:

```
> gmx mk_angndx -s protein_bb.tpr -type dihedral
-o dangle.ndx
```

7. Open **dangle.ndx** with any text editor like, for example, vim or nano, and identify the index groups that correspond to the atoms forming the φ and ψ angles. Every four consecutive atom indices present in the index groups in the **dangle.ndx** file correspond to a single dihedral angle. For example, the first index group (Titled as **[Phi=180.0_2_10.46]**) in the text box shown below (which is a sample of **dangle.ndx** file), the atom indices 2 3 4 5 correspond to the first dihedral angle and 5 6 7 8 correspond to the second dihedral angle, and so on. The atoms that contribute to the φ angles are Co_i-N_{i+1}-$C\alpha_{i+1}$-Co_{i+1} and the ψ angles are N_i-$C\alpha_i$-Co_i-N_{i+1}. When your topology contains only the backbone atoms, like in our case here, the φ angles usually have the atom indices that start with "3," which represents the first backbone carbonyl carbon atom, and the ψ angles usually have atom indices that start with "1" that represents the first backbone nitrogen atom. In the example below, the atom indices that correspond to the φ angles are highlighted in bold and the ones that correspond to the ψ angles are highlighted in Italic:

```
[ Phi=180.0_2_10.46 ]
2 3 4 5 5 6 7 8 8 9 10 11
11 12 13 14 14 15 16 17 17 18 19 20
[Phi=0.0_2_1.13]
3 4 5 6 6 7 8 9 9 10 11 12
12 13 14 15 15 16 17 18 18 19 20 21
```

```
[Phi=0.0_3_1.76]
3 4 5 6 6 7 8 9 9 10 11 12
12 13 14 15 15 16 17 18 18 19 20 21
[Phi=180.0_1_1.88]
1 2 3 4 4 5 6 7 7 8 9 10
10 11 12 13 13 14 15 16 16 17 18 19
[ Phi=180.0_2_6.61 ]
1 2 3 4 4 5 6 7 7 8 9 10
10 11 12 13 13 14 15 16 16 17 18 19
[Phi=180.0_3_2.30]
1 2 3 4 4 5 6 7 7 8 9 10
10 11 12 13 13 14 15 16 16 17 18 19
```

8. Delete all the other index groups and combine the ones that correspond to φ and ψ angles that were identified, you can also rename the group name for convenience:

```
[Psi and Phi]
3 4 5 6 6 7 8 9 9 10 11 12
12 13 14 15 15 16 17 18 18 19 20 21
1 2 3 4 4 5 6 7 7 8 9 10
10 11 12 13 13 14 15 16 16 17 18 19
```

9. Extract the dihedral angles from the backbone trajectory file:

```
> gmx angle -f protein_bb.xtc -n dangle.ndx
-or dangle.trr -type dihedral
```

10. Note the number of atom positions that are filled with cos/sin of the angles:

```
There are 12 dihedrals. Will fill 8 atom po-
sitions with cos/sin
```

and create an index file (referred to in the following steps as "covar.ndx") containing indices from 1 to the number of positions that are filled with cos/sin:

```
[Covar]
1 2 3 4 5 6 7 8
```

11. Make a reference structure for constructing the covariance matrix using the dihedral angles, you can choose any frame for the "**-e**" flag:

```
> gmx trjconv -f dangle.trr -s protein_
bb.tpr -o resized.gro –n covar.ndx -e 100
```

12. Perform the covariance analysis:

```
> gmx covar -f dangle.trr -n covar.ndx -ascii
-xpm -nofit -nomwa -noref -nopbc -s resized.
gro
```

13. Since the first two eigenvectors (Principal Components) contain the largest variance in data, we will use them for the analysis by calculating the Potential of Mean Force (PMF) at every

time frame and projecting the vectors on each other to obtain a 2D free energy landscape.

```
> gmx anaeig -v eigenvec.trr -f dangle.
trr -s resized.gro -first 1 -last 2 -2d
2dproj_1_2.xvg
```

14. Convert the 2D plot into a 3D Gibbs free energy landscape by Bolzmann invertion:

```
> gmx sham -f 2dproj_1_2.xvg -notime
-bin bindex-1_2.ndx -lp prob-1_2.xpm -ls
gibbs-1_2.xpm -lsh enthalpy-1_2.xpm -g shamlog-1_2
-lss entropy-1_2.xpm
```

Any of the *.xpm files can be converted to *.eps using the GROMACS command "gmx xpm2ps"

15. As explained in the introduction of Subheading 3 (*see* also Fig. 1), our aim is to obtain 30 representative peptide conformations from the MD simulations. For this, we will use the 30 biggest bins (conformational states) identified by the PCA analysis. In order to identify the largest bins, check the "**shamlog-1_2.log**" file that contains the energy of each bin estimated by the Boltzmann Inversion method (by **gmx sham** in the previous step). Therefore, the size of the bin is inversely proportional to its energy. Bin indices sorted by their energies are listed after the "**Minima sorted after energy**" line in the "**shamlog-1_2.log**" file (with the lowest energy bin—the most populated one—having an energy of 0). This part is shown below where the energies are highlighted in bold and the bin indices are highlighted in Italic:

```
...
...
Minima sorted after energy
Minimum 0 at index 249 energy 0.000
Minimum 1 at index 766 energy 0.101
Minimum 2 at index 754 energy 0.884
Minimum 3 at index 918 energy 1.621
Minimum 4 at index 570 energy 1.669
Minimum 5 at index 594 energy 2.533
...
...
```

Note down the indices of the 30 lowest energy bins (Minimum 0 to 29).

16. From the "**bindex-1_2.ndx**" note down (any) one frame number from each of the 30 bins whose indices were sought in the previous step. In all our test cases, the first frame that is listed under the bin index was used. Samples from the "**bindex-1_2.ndx**" are shown below where the first three

structures in the biggest bins: 249, 766, and 570 are shown and the first frame in these respective bins is highlighted in bold:

```
...
...
[249]
232
233
235
...
...
[766]
121
122
159
...
...
[570]
141
149
151
...
...
```

17. Extract the selected frames from the combined trajectory. Please note that you should extract the frames from the trajectory that contains ALL atoms and not just the backbone atoms that were used to perform PCA. Also, the numbers noted in the previous steps are the frame numbers. These need to be multiplied by 50 (time frequency in ps at which coordinates were saved in the production MD—change this value if you have modified the save frequency in the parameter files) to obtain the time stamp. In the following example, we used frame number 219 as an example:

    ```
    > gmx trjconv -f protein_nocap.xtc -s protein_
    nocap.tpr -dump 10950 -o frame_219.pdb -pbc
    mol
    ```

    When prompted, choose the index group 1, containing all atoms.

    Repeat this step to extract a total of 30 conformations (or less). We do not recommend using too many conformations for ensemble docking since it might lead to under-sampling of each conformation during the docking (a "dilution" problem), as the number of rigid-body docking models generated in HADDOCK is fixed.

Steps 9–17 can be automated by using the **dpca.sh** script that is given in the supplementary material. Ensure that the filenames are identical to what has been described in **steps 1–8** and the script should be run in the directory where all the files generated in these steps are present.

3.4 Docking and Scoring

3.4.1 Introduction to Docking Using HADDOCK v2.2

We provide here a brief introduction into the various docking and refinement stages of HADDOCK and its scoring functions. Before using a locally installed version of HADDOCK, it is recommended that the user also reads the HADDOCK manual (http://www.bonvinlab.org/software/haddock2.2/manual.html).

Initial Rigid Body Docking Stage (it0)

The initial stage of HADDOCK consists of a randomization of the starting orientations of the various molecules, followed by rigid body energy minimization (it0). During the randomization stage, the molecules are separated by a minimum of 25 Å and randomly rotated around their respective center of mass and translated within a 10 Å cube. During the following energy minimization, the molecules are treated as rigid bodies, i.e., all molecular bonds, angles, and internal rotations around bonds are frozen. The energy function being minimized contains the interaction restraints and the intermolecular van der Waals and electrostatics potentials. Typically, between 1000 (default) and 10,000 models are written to disk at this stage. The rigid body energy minimization is repeated multiple times internally (5 by default) with symmetrical solutions being sampled and minimized and only the best solution based on the HADDOCK score (see below) being written to disk. Typically, the top few hundred solutions (by default 200) are selected for further refinement.

Semi-flexible Refinement Stage (it1)

The second stage in HADDOCK consists of a semi-flexible refinement by high-temperature molecular dynamics in torsion angle space (it1), during which increasing flexibility is introduced in the interface of the complex. It consists of four stages:

1. High temperature rigid body dynamics (hot).
2. Rigid-body simulated annealing (cool1).
3. Semi-flexible simulated annealing with flexible side-chains at the interface (cool2).
4. Semi-flexible simulated annealing with fully flexible backbone and side-chains at the interface (cool3).

During this stage, the interface of the complex model is optimized. The flexible regions are defined by default automatically for all residues within 5 Å of a partner molecule (the user can, however, also define these manually). Usually, all the solutions at this stage are passed to the final refinement stage in explicit solvent.

Explicit Solvent Refinement Stage (water)

The last stage is a flexible refinement in explicit solvent (TIP3P water by default, but DMSO is also supported as a lipid environment mimic). The models from the previous stage are solvated in an 8.5 Å solvent shell and further refined using molecular dynamics simulations in Cartesian space, with weak position restraints on backbone atoms outside the interface, followed by a final energy minimization.

Scoring Functions of HADDOCK

The HADDOCK score is a weighted sum of various energy and other scoring terms. It is used to rank the models at the various stages of the docking. The weight of various terms differs for each stage of the docking process. By default, the HADDOCK score is calculated as follows (a combination that has been shown to be successful for various types of systems, from protein-protein to protein-nucleic acid complexes):

$$it0: \quad E = 0.01E_{\text{air}} + 0.01E_{\text{vdw}} + 1.0E_{\text{elec}} + 1.0E_{\text{desolv}} - 0.01\ \text{BSA}$$

$$it1: \quad E = 0.1E_{\text{air}} + 1.0E_{\text{vdw}} + 1.0E_{\text{elec}} + 1.0E_{\text{desolv}} - 0.01\ \text{BSA}$$

$$\text{water}: \quad E = 0.1E_{\text{air}} + 1.0E_{\text{vdw}} + 0.2E_{\text{elec}} + 1.0E_{\text{desolv}}$$

where E_{air} is the ambiguous interaction restraints energy, E_{vdw} is the intermolecular van der Waals energy described by a 12-6 Lennard-Jones potential, E_{elec} is the intermolecular electrostatic energy described by a Coulomb potential, E_{desolv} is an empirical desolvation energy term [28], and BSA is the buried surface area in Å^2. The nonbonded energies are calculated using an 8.5 Å cutoff using the OPLS force field parameters [29]. Additional energy terms can be included in the scoring function if other restraint types are used.

3.4.2 *Docking Protocol*

As described in Subheadings 3.1–3.3, various peptide conformations have been obtained, either built in ideal conformations, or through MD simulations followed by clustering by dihedral PCA, from which we selected 30 representatives. This ensemble of peptide conformations (33 in total) along with the unbound receptor structure will be used for docking. The following protocol describes the steps to perform docking using a local version of HADDOCK. As for the MD part, we use the 1CZY case (*see* Table 1) as an example.

Defining Ambiguous Interaction Restraints (AIRs)

Since HADDOCK is an information-driven docking approach, it is necessary to define information about the putative interfaces (note that HADDOCK does offer various ab initio docking protocols, although these will not be described here). For a manual run, this means generating an ambiguous interaction restraints (AIRs) file (the webserver will simply take a comma-separated list of residue numbers). In order to generate AIRs, it is necessary to define active and passive residues at the interface of each molecule based on experimental data and/or bioinformatics predictions. In this example, we will assume that we know the peptide-binding site on the receptor—defining it from the known complex as all residues on the receptor that are within a 5 Å distance from the peptide. This represents of course an ideal situation. In case there is no available experimental information about interface residues, various bioinformatics predictors, including our webserver CPORT (http://haddock.science.uu.nl/services/CPORT/), can help predict interface residues. HADDOCK distinguishes between active and passive

residues: Active residues must be in the interface (i.e., make contacts to either passive or active residues of the other molecule), otherwise an energy penalty will be paid; passive residues, in contrast, can be part of the interface, but are not penalized if otherwise. While active residues are typically selected based on high-quality experimental data or bioinformatics predictions, passive residues can be defined as the surface neighbors of active residues, or as a user-defined surface patch. In this protocol, we will define all residues of the peptide as passive. For our example 1CZY, the protein active residues are:

```
"41,42,44,60,62,66,67,68,77,114,115,116,117,
120,121,122,123,131,132,133,134,135,136,137,"
```

and the peptide passive residues:

```
"1,2,3,4,5,6,7."
```

After obtaining the list of active and passive residues for each molecule, the webserver for generating ambiguous interaction restraints can be used to generate the AIRs file required for HADDOCK. It can be found at http://www.bonvinlab.org/software/Haddock2.2/generate_air.html. A description of the format of the various restraints files in HADDOCK can be found in Box 4 of the original HADDOCK server article [17]. In the section "Active residues for 1st molecule" and "Passive residues for 2nd molecule," fill in the 1CZY protein active residues and peptide passive residues, respectively. In the section "Segid of 1st molecule:" and "Segid of 2nd molecule:", specify the segment IDs to use for each molecule during the docking (typically A and B). Then move to the bottom of the page, click on "Generate AIR restraints" to generate AIRs. You should be redirected to a new page that contains all the AIRs. Save this page as a restraint file "**restraints.tbl**" and copy it into your working directory. The "**restraints.tbl**" file will be used when setting up your new HADDOCK project.

Determining the Protonation State of Histidine Residues

In general, small charge differences can have a strong impact on the results of the docking. It is therefore advisable to define the protonation state of Histidine residues prior to docking (if not, by default Histidines will be considered positively charged). The HADDOCK webserver can do that automatically, using MolProbity [14] to guess the most plausible Histidine protonation states. Here, we provide a Python script that uses the "reduce" executable of MolProbity to assign the protonation state. Run the script on the pdb file(s) to determine the protonation states of Histidines:

```
> python ./molprobity.py ./protein.pdb
```

The *reduce* program will be used to add hydrogen atoms and perform basic validation checks and optimizations on the protein structure to generate a temporary new structure. Based on the presence and location of the hydrogen atoms in a Histidine residue, the script will determine its protonation state. For each histidine, if

atom HD1 and HE2 exist, the doubly protonated, charged histidine HIS+ state is assigned, if HD1 exists but not HE2, a singly protonated histidine HISD is assigned, and if HE2 exists but not HD1, a singly protonated histidine HISE is assigned.

The output of the script is shown below for the unbound form of 1CZY:

```
## Executing Reduce to assign histidine pro-
tonation states
## Input PDB: protein.pdb
HIS (73) --> HISD
HIS (109) --> HISE
```

Save this information since it will be used later while editing the run parameters.

Starting a New Project

To start a new project, it is necessary to generate the "**new.html**" file that contains the information about all the required input data for the docking. An online tool to prepare this file is available on the HADDOCK webpage: http://www.bonvinlab.org/software/Haddock2.2/start_new.html.

Three sections need to be filled. In the first section "HADDOCK and project setup," you should define the path to your local HADDOCK installation under "Current HADDOCK program directory" and the path to where your project will be created in "Path of the new project." Then, define the "Run number," 1 in this case, and the "Number of molecules for docking (max. 6)," 2 in this particular case.

In the second section, "Define the various molecules for docking", set "PDB file of 1st molecule" to the absolute path of the protein structure, "PDB file of 2nd molecule" to the absolute path of the peptide (use one of the peptide PDB files for this), use the default values for "Segid of 1st molecule" and "Segid of 2nd molecule" (unless of course you changed those values when creating the AIR file). Since we will make use of an ensemble of peptide conformations for the docking, a list file containing all absolute (or relative) paths of those conformations (3 extreme + 30 MD conformations) should be created, and set "file list for 2nd molecule (opt.)" to the absolute path of this list file. Note that in principle, instead of absolute paths, relative paths can also be used, e.g., "./peptide.pdb."

In the last section "Define the various restraint files," the "AIR restraints" should be set to the path of the "**restraints.tbl**" file generated in Subheading "Defining Ambiguous Interaction Restraints (AIRs)."

Move to the bottom of the page and click on "Save updated parameters." You should see a new webpage containing the generated file. Save it as the "new.html" file, and then copy it into your working directory.

An example of the "**new.html**" file for 1CZY (also provided in supplementary material) is given below:

```
<html>
<head>
<title>HADDOCK - start</title>
</head>
<body bgcolor=#ffffff>
<h2>Parameters for the start:</h2>
<BR>
<!-- HADDOCK -->
HADDOCK_DIR=/home/software/haddock/
haddock2.2<BR>
N_COMP=2<BR>
PDB_FILE1=../pdb_files/protein.pdb<BR>
PDB_FILE2=../pdb_files/MD_conformations/md_1.
pdb<BR>
PDB_LIST2=../pdb_files/structures.list<BR>
PROJECT_DIR=./<BR>
PROT_SEGID_1=A<BR>
PROT_SEGID_2=B<BR>
AMBIG_TBL=./restraints.tbl<BR>
RUN_NUMBER=1<BR>
submit_save=Save updated parameters<BR>
</h4><!-- HADDOCK -->
</body>
</html>
```

Now go to your working directory, make sure that the "**new.html**" file is present and then type the command (which should be defined after proper installation of HADDOCK and sourcing of the configuration script):

```
> haddock2.2
```

A new directory "runX" (X is a number that is defined as "Run number" above) will be created, containing several subdirectories and the important "**run.cns**" parameter file, which will need to be edited in the next step. For details on the various subdirectories, please refer to the online HADDOCK manual at http://www.bonvinlab.org/software/Haddock2.2/start_new_help.html.

Setting the Docking Parameters

It is necessary to modify some docking parameters in the freshly generated "**run.cns**" file. For this, the user can use an online tool or edit the file directly with a text editor. The easiest way to modify this file is to use our online tool: In the http://www.bonvinlab.org/software/Haddock2.2/Haddock-start.html page (best viewed with Firefox or Google Chrome), upload your "**run.cns**" file and click on "Edit file." You will be redirected to a new webpage that contains all parameters to run HADDOCK. For details on all the parameters, see the online manual (http://www.bonvinlab.org/

software/Haddock2.2/run.html). Below, we will deal with only the parameters that should be adjusted to run our protein-peptide protocol.

Histidine patches: These parameters are used to define the protonation state of Histidines. By default, a Histidine is doubly protonated and thus positively charged in HADDOCK. The histidine parameters only need to be defined when a histidine should be singly protonated (HISD or HISE). The information on the protonation state of the various Histidines obtained in Subheading "Determining the Protonation State of Histidine Residues," will be used here. For our example 1CZY in that section, Histidine 73 should be in HISD state and Histidine 109 in HISE. In the section, "Patch to change doubly protonated HIS to singly protonated histidine (HD1)," set the first residue of "molecule (Protein) A" to 73. Then in "Patch to change doubly protonated HIS to singly protonated histidine (HE2)," set the first residue of "molecule (Protein) A" to 109. In this particular example, the peptide does not contain any Histidine; otherwise, the same procedure should be followed for the second molecule.

Definition of fully flexible segments: This section defines the segments that are defined as fully flexible during all stages of it1. In this protocol, because of the intrinsic high flexibility of peptides, we will define all residues of the peptide as fully flexible. Therefore, set the "Start Residue" to 1 and the "End Residue" to the residue number of the last residue in the peptide, here should be 7 for the case 1CZY.

Topology and parameter files: The linkage file in this section allows defining the charged states of the N- and C-termini of the protein and peptide. If the protein or peptide is a fragment of a larger protein or was capped in experiments for some specific reason, the N- and/or C-terminus should be uncharged. In HADDOCK, the default linkage file used to generate the topology is "protein-allhdg5-4.link," which produces charged N and C termini. For uncharged termini, the linkage file "protein-allhdg5-4-noter.link" should be used. For uncharged N-terminus and charged C-terminus, use "protein-allhdg5-4-noNter.link"; for charged N-terminus and uncharged C-terminus, use "protein-allhdg5-4-noCter.link."

For our particular example 1CZY, we will use "protein-allhdg 5-4-noter.link" for the peptide since the peptide is a fragment of a larger protein and capped in its N-terminus.

Number of structures to dock: Due to the flexibility of the peptide, the number of decoys to generate should be increased to improve the sampling of all conformations of the protein-peptide complex. Since we used 33 peptide conformations (3 extreme conformations + 30 MD cluster representatives) as initial ensemble for docking, we will change the "number of structures for rigid body docking" from

1000 to 9900 (in that way each starting conformation is used 300 times), and the "number of structures for refinement" from 200 to 400.

Number of MD steps in the docking protocol: To improve the sampling of protein-peptide interactions and in particular to allow the peptide to better sample its conformation in the context of the receptor, the number of MD steps for the it1 stages also needs to be increased: From 500/500/1000/1000 to 2000/2000/4000/4000 for the hot, cool1, cool2, and cool3 stages in it1, respectively.

Final explicit solvent refinement: Just like the number of structures to dock, the "number of structures for the explicit solvent refinement" is increased from 200 to 400.

Analysis and clustering: The "clustering method" is set to RMSD, and due to the smaller size of peptide the "RMSD cutoff for clustering" is decreased to 5 Å.

Parallel jobs: The user should specify the local "queue command" (e.g., simply csh if using a single computer, or a batch queue submission command for a cluster (e.g., qsub)), the absolute path of "cns executable" and define for "cpunumber" a number that matches the number of cores on the system (or the number of allocated slots in the queue in the batch system). The internal job dispatching routines will use this setting to limit the number of concurrent refinement jobs. Note that for rigid-body stage jobs, given the computational efficiency of this algorithm, several individual minimizations are bundled together in each job.

After updating the parameters above, click "Save updated file" on the page bottom, and then save the file as a new run.cns and copy it to the runX directory to replace the old one.

The following lines describe how these changes would look like if done manually simply in a text editor:

Histidine patches:

```
A_hisd_resid_1=73;
A_hise_resid_1=109;
```

The histidine protonation states are defined by these parameters.

Definition of fully flexible segments:

```
B_start_fle_1="1";
B_end_fle_1="7";
```

The parameters define the fully flexible segments in it1 step.

Topology and parameter files:

```
prot_link_B="protein-allhdg5-4-noter.link";
```

The parameter sets the charged states of N-terminus and C-terminus of the molecule. Here, it sets both termini uncharged for the peptide of 1CZY.

Number of structures to dock:

```
structures_0=9900;
structures_1=400;
```

These parameters set the number of structures to dock for it0 and it1.

DOCKING protocol:

```
initiosteps=2000;
cool1_steps=2000;
cool2_steps=4000;
cool3_steps=4000;
```

These parameters set the number of MD steps for hot, cool1, cool2, and cool3 stage in it1 step.

Final explicit solvent refinement:

```
waterrefine=400;
```

The parameter sets the number of structures to dock for the water step.

Analysis and clustering:

```
clust_meth="RMSD";
clust_cutoff=5;
```

These parameters set the clustering method and its cutoff.

Parallel jobs:

```
queue_1="csh";
cns_exe_1="/home/software/bin/cns";
cpunumber_1=50;
```

These parameters set the local queue command, path of cns program, and the number of parallel jobs.

Start the Docking

After ensuring that all parameters have been properly defined as explained in the previous steps, navigate to the runX directory and launch the docking by typing:

```
> haddock2.2 &>HADDOCK.log &
```

The docking should start in background and the information about the run will be written to the HADDOCK.log file. During the docking process, HADDOCK writes docking decoys in PDB format and outputs the ranked PDB files in file.cns, file.list and file.nam files at the end of each docking step in runX/structures/it0, runX/structures/it1 and runX/structures/it1/water directories, respectively. The file.cns, file.list and file.nam files contain a list of generated structures sorted on HADDOCK score. For it1 and water stages, the generated structures are automatically analyzed and the results are placed in the runX/structures/it1/analysis and runX/structures/it1/water/analysis directories, respectively.

3.4.3 Analysis

Automatic Analysis

As described above, the results of automatic analysis for it1 and water steps are placed in the analysis directories under each directory, respectively. Here, we will describe some relevant files containing useful information.

`fileroot_ave.pdb and fileroot_X.pdb:`

These are the models in PDB format. fileroot_ave.pdb is the average structure generated by superimposing the structures of docking solutions on the backbone atoms of interface residues, while the superimposed models are fileroot_N.pdb (N is a number that corresponds to the ranking of the model in the file.list file). The interface residues are automatically determined from an analysis of all generated models. Note that the average model might not be of much relevance in cases where very different solutions are sampled.

`fileroot_rmsd.disp:`

This file contains the pairwise RMSD matrix calculated over all models. The RMSD calculated here is the ligand interface RMSD, i.e., the structures are fitted on backbone atoms of interface residues of the first molecule and the RMSD is calculated on the interface backbone atoms of the second molecule. This file is used as input for the RMSD clustering. If the clustering method defined in run.cns is FCC (fraction of common contacts) instead, the name of this file will become fileroot_fcc.disp. and contain the fraction of common contacts between models [30].

`cluster.out:`

The file contains the list of clusters generated based on the matrix in the fileroot_rmsd.disp or fileroot_fcc.disp file, depending on the clustering method used. The clusters are numbered according to the size of the cluster, e.g., the largest cluster is cluster 1. The cluster.out file is used as input for analysis of clusters (ana_cluster.csh) described below.

`energies.disp, edesolv.disp and ene-reside.disp:`

These files contain various energy terms. The bonded and nonbonded energies and buried surface area for each structure are written to energies.disp, together with the average values over the ensemble. The empirical desolvation energies are contained in edesolv.disp. The ene-residue.disp file lists the per-residue intermolecular energies for all interface residues.

`hbonds.disp, ana_hbonds.lis and nbcontacts.disp, ana_nbcontacts.lis:`

The hbonds.disp file contains the intermolecular hydrogen bonds for each model, while the ana_hbonds.lis file lists all hydrogen bonds with their occurrence and average distance.

Similar information for intermolecular hydrophobic contacts is provided in nbcontacts.disp and ana_nbcontacts.lis.

`geom.disp:`

The file contains the averaged deviations from ideal covalent geometry (bonds, angles, impropers, and dihedrals) for each structure and averaged over all structures.

`noe.disp:`

The file contains the number of distance restraints violations per structure and averaged over the ensemble over all distance restraint classes and for each class (unambiguous, ambiguous, hbonds). Similar files are generated for dihedral angle restraints (dihedrals.disp), residual dipolar coupling restraints (sani.disp), intervector projection angle restraints (vean.disp), diffusion anisotropy restraints (dani.disp), and pseudo contact shifts restraints (pcs.disp).

`ana_XXX.lis:`

These files report restraint violations over the ensemble of models, giving the number of times various restraints are violated, the average distance, and the violation per restraint. The XXX can be dihed_viol, dist_viol_all, hbond_viol, noe_viol_all, noe_viol_ambig, and noe_viol_unambig.

Manual Analysis

Besides the automatic analysis, the user should also perform manual analysis of the models and clusters. For this purpose, a number of scripts are provided in the runX/tools directory.

1. *Collecting model statistics using ana_structures.csh*: This script extracts information from the header of the PDB files such as various energy terms, violation statistics, and buried surface area and calculates the overall backbone RMSD of each structure superimposed on the top ranking model. To run it type:

   `>$HADDOCKTOOLS/ana_structures.csh`

   in the runX/structures/it1 or runX/structures/it1/water directory.
   It generates a number of "file.nam_XXX" and "structures_XXX-sorted.stat" files (XXX is energy term). The "file.nam_XXX" file contains the values of the respective energy (or other) term XXX for all structures. All of these terms are combined into one file and sorted in different ways, generating the corresponding "structures_XXX-sorted.stat" files. Of these, structures_haddock-sorted.stat is usually the most important, which corresponds to the HADDOCK score ranking.
   Relationships between these energy terms can be checked by plotting. For this purpose the make_ene-rmsd_graph.csh script

is provided. For example, the user can make a plot of the HADDOCK score as a function of the RMSD:

```
> $HADDOCKTOOLS/make_ene-rmsd_graph.csh 3 2
structures_haddock-sorted.stat
```

It will generate a ene_rmsd.xmgr file in xmgr format which can be displayed with xmgr or xmgrace:

```
> xmgrace ene_rmsd.xmgr
```

2. *RMSD clustering using cluster_struc*: This program is used in HADDOCK to perform clustering based on RMSDs. In the process of automatic analysis, if RMSD clustering was defined in run.cns, it has been run automatically in each analysis directory. However, the user can run it again to try different clustering cutoffs depending on the complex studied. It takes the fileroot_rmsd.disp file as input:

   ```
   > $HADDOCKTOOLS/cluster_struc [-f] fileroot_
   rmsd.disp cutoff min_cluster_size>cluster.
   out
   ```

   Here, the -f is an option for full linkage clustering algorithm (not used by default), the cutoff is the RMSD cutoff used to determine if two structures belong in the same cluster, and min_cluster_size is the minimum number of models to define a cluster.
   The output in the cluster.out file looks like:

   ```
   Cluster 1 -> 2 4 5 9 11 12 14 20 121 127 129
   141 145 156 170
   Cluster 2 -> 1 48 51 56 58 93 96 139 161 164
   171 181 187
   Cluster 3 -> 36 7 37 49 112 148
   ```

   The numbering of the clusters is based on the size of the cluster, and the numbering of the structures corresponds to the position of the structure in the file.list file. The first structure of each cluster corresponds to the cluster center and the other structures are sorted according to their index.

3. *FCC clustering using cluster_fcc.py*: This Python script is used to perform clustering based on the fraction of common contacts (FCC) if FCC clustering was defined in run.cns. FCC is an alternative metric to measure the structural similarity between two docking models, based on the network of residue-residue interactions at the interface of the models. As for RMSD clustering, the user can choose to run it again to try different clustering cutoffs depending on the complex studied. It takes the fileroot_fcc.disp file as input:

   ```
   > $HADDOCKTOOLS/cluster_fcc.py fileroot_fcc.
   disp cutoff -c min_cluster_size > cluster.out
   ```

where cutoff is the FCC cutoff used to determine if two structures belong to the same cluster, and min_cluster_size is the minimum number of models to define a cluster.

4. *Analysis of clusters using ana_cluster.csh*: The ana_clusters.csh script calculates various statistics on a per cluster level. It takes the cluster.out file as input. To run it type:

```
>$HADDOCKTOOLS/ana_clusters.csh [-best #]
analysis/cluster.out
```

in the runX/structures/it1 or runX/structures/it1/water directory. The -best # is an optional argument to generate additional files with calculation only on the best # structures of a cluster, e.g., the top four structures of a cluster sorted on their HADDOCK score as done by default by the HADDOCK web server; this allows removing the dependency of the calculated averages on size of the various clusters.
Like the output of the ana_structures.csh script, the ana_clusters.csh script also generates a number of files containing values of different energy terms XXX but over models belonging to the same cluster (clustX), e.g., file.nam_clustX_XXX files, based on the list of models for each cluster stored in the file.nam_clustX files. The script also calculates averages of various energy terms for each cluster, which can be found in the various cluster_XXX.txt files. All these are combined and sorted in various ways in clusters_XXX-sorted.stat files. If the option "–best #" is used, additional files will be created containing the average values over the best # structures of each cluster, i.e., file.nam_clustX_best#, cluster_XXX.txt_best# and clusters_XXX-sorted.stat_best# files. Of all these files, clusters_haddock-sorted.stat and clusters_haddock-sorted.stat_best# are usually the most relevant.

5. *Rerunning automatic analysis on the basis of clusters*: After having performed the cluster-based analysis, it is possible to rerun the HADDOCK automatic analysis for a given cluster. For this the user needs to create cluster-specific files (e.g., file.cns_clustX_best#, file.list_clustX_best#, and file.nam_clustX_best#) and directory (e.g., analysis_clustX_best#). To simplify this process, the make_links.csh script is provided. To run it type:

```
>$HADDOCKTOOLS/make_links.csh clustX_best#
```

This will automatically move the original file.cns, file.list and file.nam files and analysis directory to new files and directory by adding a suffix _all, and then make links to cluster-specific files and directory, i.e.,

```
file.cns     -> file.cns_clustX_best#
file.list     -> file.list_clustX_best#
file.nam     -> file.nam_clustX_best#
analysis     -> analysis_clustX_best#
```

To rerun the analysis, go back to the runX directory and restart HADDOCK:

```
> haddock2.2
```

Once finished, the user will find a new directory analysis_clustX_best# that contains cluster-specific result files described in Subheading "Automatic Analysis."

3.5 Case Studies

Following the protocol described above, we performed unbound/unbound docking for six protein-peptide complexes from the protein-peptide benchmark [11], using a combination of ideal peptide conformations and MD cluster representatives. These six complexes (Table 1) correspond to two easy, two medium, and two hard docking cases (based on the classification of Trellet et al. [9]). The length of the peptides in these systems varies from 6 to 13 amino acids, while the proteins are much larger, varying from 74 to 214 residues.

We performed the docking using both the original three conformations (alpha-helix, polyproline-II, and extended) protocol (regular protocol) [9] and by adding 30 additional conformations sampled in MD simulations as described in this chapter (MD-based protocol). To assess the performance of the docking, the interface RMSD measure from the community-wide experiment CAPRI (Critical Assessment of PRedicted Interactions) [31, 32] is used as criteria, which is calculated on interface residues by superimposing the docking solutions to the crystal structure of bound complex. In the case of protein-peptide complexes, in CAPRI a docking solution is considered acceptable if its interface RMSD is less than 2 Å.

We summarized in Table 2, for both the regular protocol and the current MD-based protocol, the number of acceptable models out of the 400 water-refined models, together with the rank of the first acceptable model and the first acceptable cluster in the list of models or clusters sorted on HADDOCK score. This allows us to compare the docking performance of both protocols. For the easy cases, 1DDV and 1LVM, the MD-based protocol generated less acceptable models. This is due to the "dilution" problem mentioned above: with a larger number of starting conformations, only few will lead to acceptable models and accordingly the total number of acceptable models is expected to decrease depending on the information used to drive the docking. On the other hand, when large conformational changes are taking place, it seems that the MD-based protocol does improve the number of acceptable models (1CZY and 1NX1) and the ranking. However, both protocols fail for two cases, for long peptides (11 and 13 amino acids, for 1D4T and 1HC9, respectively), with rather large conformational changes. 1HC9 is especially challenging since the peptide forms a b-hairpin conformation in its bound form that is not sampled in the starting models. This clearly illustrates the challenges of

Table 2
Comparison of (A) unbound/unbound docking performance between the original three-conformations protocol and the MD-based protocol presented in this chapter and (B) interface RMSD of the best and first acceptable model using the MD-based protocol

(A)

		Number of acceptable[a] models		Rank[b] of first acceptable model		Rank[b] of first acceptable cluster[c]	
Case difficulty	**PDB ID complex**	**Regular protocol**	**MD-based protocol**	**Regular protocol**	**MD-based protocol**	**Regular protocol**	**MD-based protocol**
Easy	1DDV	39	30	11	3	5	1
	1LVM	176	92	1	1	1	1
Medium	1CZY	74	175	1	1	1	1
	1D4T	0	0	NA	NA	NA	NA
Hard	1HC9	0	0	NA	NA	NA	NA
	1NX1	58	62	6	3	3	2

(B)

Case difficulty	**PDB ID complex**	**i-RMSD (Å)/rank of best model (Å)**	**i-RMSD (Å)/rank of first acceptable[a] model**
Easy	1DDV	1.74/95	1.96/3
	1LVM	1.26/40	1.64/1
Medium	1CZY	0.93/42	1.31/1
	1D4T	2.31/3	NA
Difficult	1HC9	4.42/131	NA
	1NX1	1.28/43	1.59/3

The original protocol [9] uses three peptide conformations (alpha-helix, polyproline-II, and extended), while 30 additional conformations sampled in MD simulations were added in the MD-based protocol. Both protocols output 400 docking models at the end of the HADDOCK process

[a]A model is defined as acceptable if its interface RMSD (i-RMSD) from the reference is less than 2 Å according to the criteria of CAPRI. The i-RMSD is calculated on interface backbone atoms of docking models superimposed onto the crystal structure

[b]The ranking of the first acceptable model/cluster is the position of the first acceptable model/cluster in the list of models/clusters sorted on HADDOCK score

[c]A cluster is defined as acceptable when at least one model is acceptable within the top four models

protein-peptide docking. The best models for each case are shown in Fig. 2, superimposed onto the reference crystal structure.

In conclusion, the presented results, although taken from a limited number of cases, seem to indicate that the presented MD-based protocol is better at generating acceptable models with HADDOCK. This was however for an ideal case where the binding site on the receptor is well defined. Lack of proper information, high-flexibility, and large conformational changes still remain major challenges to be addressed in protein-peptide interaction modeling.

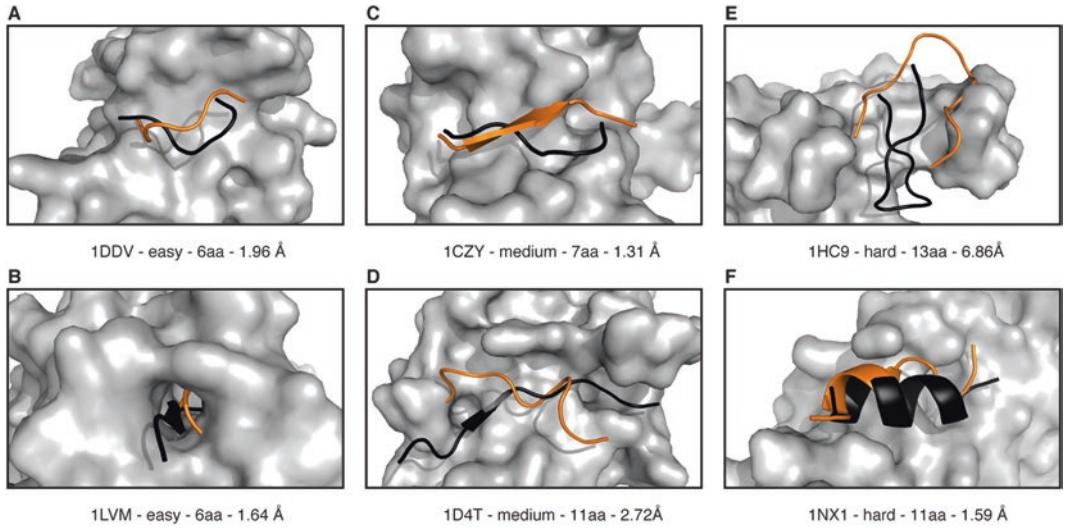

Fig. 2 View of the HADDOCK top-ranking first acceptable model for the six protein-peptide unbound/unbound docking cases using the MD-based protocol. The PDB ID as well as difficulty, peptide length and interface RMSD values are indicated for all six cases. The model selected for illustration is the acceptable model with the best rank for (**a**) 1DDV, (**b**) 1LVM, (**c**) 1CZY, and (**f**) 1NX1, and the model with the best HADDOCK score for (**d**) 1D4T and (**e**) 1HC9 (no acceptable models were generated for those two cases). The model peptide is shown in *orange* together with the reference peptide in the crystal structure of the complex in *black*. Docking model and crystal structure were superimposed on backbone atoms of the protein. The protein in the crystal structure is shown in surface representation. Figure generated with PyMOL [12]

Acknowledgments

C. Geng acknowledges financial support from the China Scholarship Council, grant NO. 201406220132. This protocol is adapted from a computer practical offered to our chemistry bachelor students [33].

References

1. Craik DJ, Fairlie DP, Liras S, Price D (2013) The future of peptide-based drugs. Chem Biol Drug Des 81:136–147
2. Otvos L (2008) Peptide-based drug design: here and now. Methods Mol Biol 494:1–8
3. Fischer E (1894) Einfluss der Configuration auf die Wirkung der Enzyme. Berichte der Dtsch Chem Gesellschaft 27:2985–2993
4. Koshland DE (1959) Enzyme flexibility and enzyme action. J Cell Comp Physiol 54: 245–258
5. Kumar S, Ma B, Tsai CJ, Sinha N, Nussinov R (2000) Folding and binding cascades: dynamic landscapes and population shifts. Protein Sci 9:10–19
6. Rubin MM, Changeux JP (1966) On the nature of allosteric transitions: implications of non-exclusive ligand binding. J Mol Biol 21: 265–274
7. Monod J, Wyman J, Changeux J-P (1965) On the nature of allosteric transitions: a plausible model. J Mol Biol 12:88–118
8. Vogt AD, Di Cera E (2012) Conformational selection or induced fit? A critical appraisal of the kinetic mechanism. Biochemistry 51: 5894–5902
9. Trellet M, Melquiond ASJ, Bonvin AMJJ (2013) A unified conformational selection and induced fit approach to protein-peptide docking. PLoS One 8:e58769

10. Diella F et al (2008) Understanding eukaryotic linear motifs and their role in cell signaling and regulation. Front Biosci 13:6580–6603
11. London N, Movshovitz-Attias D, Schueler-Furman O (2010) The structural basis of peptide-protein binding strategies. Structure 18:188–199
12. Pymol T, Graphics M, Schrödinger V. The PyMOL Molecular Graphics System, Version 1.5.0.4 Schrödinger, LLC. 5
13. Abraham MJ et al (2015) GROMACS: high performance molecular simulations through multi-level parallelism from laptops to supercomputers. SoftwareX 1:19–25
14. Chen VB et al (2010) MolProbity: all-atom structure validation for macromolecular crystallography. Acta Crystallogr D Biol Crystallogr 66:12–21
15. Dominguez C, Boelens R, Bonvin AMJJ (2003) HADDOCK: a protein-protein docking approach based on biochemical or biophysical information. J Am Chem Soc 125: 1731–1737
16. de Vries SJ et al (2007) HADDOCK versus HADDOCK: new features and performance of HADDOCK2.0 on the CAPRI targets. Proteins 69:726–733
17. de Vries SJ, van Dijk M, Bonvin AMJJ (2010) The HADDOCK web server for data-driven biomolecular docking. Nat Protoc 5:883–897
18. van Zundert GCP et al (2015) The HADDOCK2.2 Web Server: user-friendly integrative modeling of biomolecular complexes. J Mol Biol. doi:10.1016/j.jmb.2015.09.014
19. Brünger AT et al (1998) Crystallography & NMR system: a new software suite for macromolecular structure determination. Acta Crystallogr D Biol Crystallogr 54:905–921
20. Brunger AT (2007) Version 1.2 of the crystallography and NMR system. Nat Protoc 2: 2728–2733
21. Mitternacht S (2016) FreeSASA 0.6.2: Solvent accessible surface area calculation. doi:10.5281/zenodo.44748
22. Rodrigues JPGLM, Bonvin AMJJ (2014) Integrative computational modeling of protein interactions. FEBS J 281:1988–2003
23. Karaca E, Bonvin AMJJ (2013) Advances in integrative modeling of biomolecular complexes. Methods 59:372–381
24. Melquiond ASJ, Bonvin AMJJ (2010) Data-driven docking: using external information to spark the biomolecular rendez-vous. In: Zacharias M (ed) Protein-protein complexes analysis, modeling and drug design. Imperial College Press, Munich, pp 183–209. doi:10.1142/9781848163409_0007
25. de Vries SJ, Bonvin AMJJ (2008) How proteins get in touch: interface prediction in the study of biomolecular complexes. Curr Protein Pept Sci 9:394–406
26. de Vries SJ, Bonvin AMJJ (2011) CPORT: a consensus interface predictor and its performance in prediction-driven docking with HADDOCK. PLoS One 6:e17695
27. Lindorff-Larsen K et al (2010) Improved side-chain torsion potentials for the Amber ff99SB protein force field. Proteins 78:1950–1958
28. Fernández-Recio J, Totrov M, Abagyan R (2004) Identification of protein-protein interaction sites from docking energy landscapes. J Mol Biol 335:843–865
29. Jorgensen WL, Tirado-Rives J (1988) The OPLS [optimized potentials for liquid simulations] potential functions for proteins, energy minimizations for crystals of cyclic peptides and crambin. J Am Chem Soc 110: 1657–1666
30. Rodrigues JPGLM et al (2012) Clustering biomolecular complexes by residue contacts similarity. Proteins 80:1810–1817
31. Janin J (2005) Assessing predictions of protein-protein interaction: the CAPRI experiment. Protein Sci 14:278–283
32. Lensink MF, Wodak SJ (2013) Docking, scoring, and affinity prediction in CAPRI. Proteins 81:2082–2095
33. Rodrigues JPGLM, Melquiond ASJ, Bonvin AMJJ (2016) Molecular Dynamics Characterization of the Conformational Landscape of Small Peptides: A series of hands-on collaborative practical sessions for undergraduate students. Biochemistry and Molecular Biology Education 44:160–167