M1 ISDD - BI

PROTEIN DOCKING

February 2018

PRACTICAL SESSION 1

(/home/sdv/all/protdocking/practical_1/)

Juan Fernandez-Recio

CSIC, BSC

juanf@bsc.es

1. INTRODUCTION

The goal of this practical is to use rigid-body docking to predict the structure of a protein-protein complex from the coordinates of the unbound subunits, based on geometry complementarity and FFT-based search.

Why predicting protein-protein interactions?

Most of protein functions in biological processes involve the interaction with other biomolecules and the formation of specific protein-protein complexes. Yet obtaining structural information on these complexes is technically difficult. Computer simulations can help to predict the geometry of a protein-protein complex and to understand the mechanism of protein association.

Which docking program can I use?

There are several computer tools publicly available for protein-protein docking. Here we will learn about two of the most popular rigid-body docking programs *FTDOCK* (http://www.sbg.bio.ic.ac.uk/docking/ftdock.html) and *ZDOCK-2.1* (http://zlab.bu.edu/zdock/), which are very fast and can be freely used for academic research. The programs use a Fast Fourier Transform (FFT) protocol to speed up the translational sampling of two proteins, in search of orientations with optimal shape complementarity. Finally, we can use graphics tools such as *PyMol*, *VMD* or *ICM* to visualize and analyze the docking solutions.

2. SOFTWARE NEEDED

All needed software is pre-installed in:

```
/home/sdv/all/protdocking/software/
```

2.1. ZDOCK

You will use the ZDOCK 2.1 linux installation in the classroom. For other installations, you can find further details and other installations in this web site:

```
http://zdock.umassmed.edu/software
```

All needed binaries are installed in:

```
/home/sdv/all/protdocking/software/IntelP3_Linux/
```

For more convenient use, you can define the following environment variable:

```
export ZDOCK=/home/sdv/all/protdocking/software/IntelP3_Linux/
```

IMPORTANT: this variable will be valid in your current shell ONLY for your current session. So, for each new session (i.e. new open shell), you have to define this environment variable again. If you want this variable to be defined by default in any session, you should include the above command in your .bashrc file.

Before running ZDOCK commands, you need to copy the following files in your working directory:

```
cp $ZDOCK/uniCHARMM ./
cp $ZDOCK/create_lig ./
```

WARNING: before running zdock, type the following command in your current shell (or in your .bashrc):

```
export _POSIX2_VERSION=199209
```

(otherwise ZDOCK would not work on the classroom machines)

ZDOCK can be run by:

```
/home/sdv/all/protdocking/software/IntelP3_Linux/zdock
```

Or if you previously defined the ZDOCK variable:

```
$ZDOCK/zdock
```

2.2. FTDock

You will use the FTDock 2.0 linux installation in the classroom. For other installations, you can find further details and instructions in the following web site:

```
http://www.sbg.bio.ic.ac.uk/docking/download.html
```

All needed binaries are installed in:

```
/home/sdv/all/protdocking/software/3D_Dock/progs/
```

For more convenient use, you can define the following environment variable:

```
export FTDOCK=/home/sdv/all/protdocking/software/3D_Dock/progs/
```

You can run FTDock by:

```
/home/sdv/all/protdocking/software/3D_Dock/progs/ftdock
```

Or if you previously defined the FTDOCK variable:

```
$FTDOCK/ftdock
```

2.3. Installation of ICM for Linux (OPTIONAL)

We will use the ICM-Browser 3.8.6a installation in the classroom computers. For other installations you can find further details and instructions in www.molsoft.com.

In order to run ICM with the proper libraries, we need to enter here:

```
ssh -X $USER@localhost
```

From there, ICM can be called simply by:

```
icm
```

(this will call a shell script that automatically defines the `MOLBROWSERPROHOME` variable, and launches the GUI version)

DISCLAIMER: *Please be aware that the installation files provided here are for purpose of this tutorial ONLY. For any other use, you should download the files from the ICM web page (http://www.molsoft.com) and register for a proper license (free for academics).*

3. DOCKING BARNASE AND BARSTAR

Barnase is an extracellular ribonuclease. Barstar is an intracellular inhibitor of barnase. To stop barnase working inside the cell, barstar binds to barnase very tightly (high affinity) and very quickly (high association rate). The binding of barnase and barstar has been very well characterized experimentally in terms of structure, energetics and kinetics. The interacting surfaces have a good surface and electrostatics complementarity, so it's not surprising that the binding is very strong and fast. That makes them a perfect target for protein-protein docking studies. We will use here the unbound structures of barnase (PDB code 1A2P) and barstar (PDB code 1A19). The structure of the barnase-barstar complex is known (PDB code 1BRS), so we can compare the docking results with the experimental data.

PDB structures:

Unbound barnase: code 1A2P (chain A)

Unbound barstar: code 1A19 (chain A)

Reference barnase/barstar complex: code 1BRS (chains A,D)

3.1. Prepare the input files

Download the coordinate files of the unbound proteins from the Protein Data Bank (www.pdb.org). There are several structures available. In this tutorial we will use the coordinates under the codes 1A2P and 1A19 for barnase and barstar, respectively, but it can be interesting to try other structures and compare the results.

3.1.1. Prepare files for docking.

From www.pdb.org, download coordinates of pdb code 1A2P and save them as `1A2P.pdb`. First we need to clean this file. We need to remove the lines starting with 'ANISOU' (*anisotropic temperature factors*, which may cause ZDOCK not to work correctly). You can use the following linux command to make a file (`1A2Patom.pdb`) that keeps only the lines starting with 'ATOM':

```
cat 1A2P.pdb | grep ATOM > 1A2Patom.pdb
```

Now this file will contain three symmetric copies of barnase, but we need only one of the monomers for docking. Thus, you should edit the pdb file (with a standard text editor such as *vi* or *emacs*) to keep only the first chain A (water molecules and other ions can also be removed), and we will save it as `1A2P_A.pdb`.

Download coordinates of pdb code 1A19 and save them as `1A19.pdb`. The file will contain two monomers, and we will only use one. We will save coordinates of chain A in the file `1A19_A.pdb` (with a standard text editor, as explained before).

3.1.2. Prepare a pdb file with the coordinates of the complex to evaluate the docking results

Download coordinates of pdb code 1BRS and save them as `1BRS.pdb`. The file will contain three copies of the barnase/barstar complex (the first copy is formed by molecules A and D for barnase and barstar, respectively; the second copy are molecules B and E; and the third copy are molecules C and F).

We will create a `1BRS_AD.pdb` file with one of the copies of the complex formed by chains A and D (with a standard text editor, as explained before). We will use this file later to evaluate the docking results.

3.2. Running ZDOCK

There are two options: we can run ZDOCK on a web server, or locally. We will try both and will compare the results.

3.2.1. Running ZDOCK on a web server.

Upload your receptor and ligand pdb files on the web server:

http://zdock.umassmed.edu/

The results will be sent to the e-mail address indicated in the submission. This e-mail will point to a web page from which you can download the ZDOCK output file, together with the receptor and ligand files used during the docking. You can rename the ZDOCK output file, but the receptor and ligand files should not be renamed (their names are stored in the ZDOCK ouput file).

WARNING: the ZDOCK web server is usually very fast, but it might take longer time if everybody is sending jobs at the same time. We just want the results to compare them with the local run (see later). Beware that the web server and local versions might be different, and therefore the results might vary.

3.2.2. Running zdock locally

Remember that before running ZDOCK commands, we need to copy the following files in our working directory:

```
cp $ZDOCK/uniCHARMM ./
cp $ZDOCK/create_lig ./
```

3.2.2.1. Prepare the receptor and ligand files

We will create new receptor and ligand pdb files (`1BRS_r.pdb` and `1BRS_l.pdb`, respectively) with the modifications required by ZDOCK:

```
$ZDOCK/mark_sur 1A2P_A.pdb 1BRS_r.pdb
$ZDOCK/mark_sur 1A19_A.pdb 1BRS_l.pdb
```

NOTE: do not worry about some possible error messages from "unknown type", as long as you have the new receptor and ligand files just built in your directory.

3.2.2.2. Send the docking simulations

We will submit the ZDOCK job:

WARNING: before running zdock type the following in your shell:

```
export _POSIX2_VERSION=199209
```

(otherwise ZDOCK would not work on the classroom machines)

```
$ZDOCK/zdock -R 1BRS_r.pdb -L 1BRS_l.pdb -o 1BRS_zdock.out &
```

(when finished, the results will be stored in output file "1BRS_zdock.out")

NOTE: this step will take some time, between 5 and 30 min (depending on the CPU architecture). When finished, we will evaluate the results in section 4. Meanwhile continue to section 3.3

3.3. Running FTDock

3.3.2. Send the docking simulations

We will submit the FTDock job using this command in a single line:

```
$FTDOCK/ftdock -static 1A2P_A.pdb -mobile 1A19_A.pdb -noelec
 -calculate_grid 1.2 -angle_step 12 -internal -15 -surface 1.3 -keep 3
 -out 1BRS.ftdock > 1BRS.ftdock.log &
```

(when finished, the results will be stored in output file "1BRS.ftdock")

Note that we are using here FTDock with grid size 1.2 and the "no electrostatics" option, for speediness. For more accurate calculations, we would need to run FTDock with grid size 0.7 and "electrostatics on".

NOTE: this step will take some time, around 12-30 min, depending on the CPU architecture. When it is finished, we will evaluate the results in section 5. Meanwhile continue to section 4.

4. A CAPRI CASE

We will now perform docking on a real case from the CAPRI experiment [1], target T26 [2]. The goal now is to generate rigid-body docking poses with FTDock and ZDOCK for this target, which will be later scored with pyDock function, in Practical #2.

CAPRI target 26 consists on the prediction of a complex from the unbound coordinates of the proteins TolB and Pal. These proteins from E.Coli form a complex involved in maintaining the bacteria outer membrane stability.

These are the files provided by the CAPRI organizers as input for docking:

Unbound TolB: code 1C5K

Unbound Pal: code 1OAP

4.1. Prepare the input files

From www.pdb.org, save coordinates for PDB entry 1C5K as `1C5k.pdb`. Similarly, save coordinates for PDB entry 1OAP as `1OAP.pdb`. More details about the proteins will be given in Practical #2. In that tutorial we will learn to automatically prepare the input files with pyDock tool. Here we will process them manually so that we can start the docking.

First, we will keep only ATOM records:

```
cat 1C5K.pdb | grep ATOM > 1C5Katom.pdb
cat 1OAP.pdb | grep ATOM > 1OAPatom.pdb
```

Then, edit `1OAPatom.pdb` to remove last OXT atom.

In this way, we can use in Practical #2 the resulting data generated from the docking of these files defined here.

4.2. Running ZDOCK

Remember that before running ZDOCK commands, we need to copy the following files in our working directory:

```
cp $ZDOCK/uniCHARMM ./
cp $ZDOCK/create_lig ./
```

4.2.1. Prepare the receptor and ligand files

We will create new receptor and ligand pdb files (`1C5K_r.pdb` and `1OAP_l.pdb`, respectively) with the modifications required by zdock:

```
$ZDOCK/mark_sur 1C5Katom.pdb 1C5K_r.pdb
$ZDOCK/mark_sur 1OAPatom.pdb 1OAP_l.pdb
```

4.2.2. Send the docking simulations

We will submit the zdock job:

WARNING: before running the zdock job, type the following in your shell:

```
export _POSIX2_VERSION=199209
```

(otherwise ZDOCK would not work on the classroom machines)

```
$ZDOCK/zdock -R 1C5K_r.pdb -L 1OAP_l.pdb -o T26.zdock &
```

(this step will take some time, let it run and when finished, we will continue in Practical #2)

4.3. Running FTDock

We will submit the FTDock job using this command in a single line:

```
$FTDOCK/ftdock -static 1C5Katom.pdb -mobile 1OAPatom.pdb -noelec
  -calculate_grid 1.2 -angle_step 12 -internal -15 -surface 1.3 -keep 3
  -out T26.ftdock > T26.ftdock.log &
```

Note that we are using here FTDock with grid size 1.2 and the "no electrostatics" option, for speediness. For more accurate calculations, we would need to run FTDock with grid size 0.7 and "electrostatics on".

(this step will take some time, let it run and when finished, we will continue in Practical #2)

5. ANALISYS OF DOCKING SOLUTIONS

We will analyze here the docking solutions generated in section 3 (the ones in section 4 will be analyzed in the pyDock practical #2).

5.1. Create the pdb files from the ZDOCK docking output

We can create pdb files from the docking conformations in the ZDOCK output file.

README: The ZDOCK output file contains information for 2000 docking solutions, so using the unmodified output would create 2000 pdb files in your current directory. It is thus advisable to edit the ZDOCK output file to keep only the top N solutions (for instance, N=10). You can use a standard text editor for that, or the following linux command (for instance, if you wan to keep the top 10 solutions, i.e. N=10):

```
head -n 14 1BRS_zdock.out > 1BRS_zdock_10.out
```

Note that we use "-n 14" to keep also the first 4 lines (in addition to the 10 lines with the top 10 solutions), which contain important information.

Now we can create pdb files for the top 10 docking solutions from the short ZDOCK output file (pdb files will be named *complex.1*, *complex.2* and so on...):

```
$ZDOCK/create.pl 1BRS_zdock_10.out
```

We will visualize the resulting docking poses later in the section 5.3. Meanwhile, proceed to section 5.2.

5.2. Create the pdb files from the FTDock docking output

We can create pdb files from the FTDock output file:

```
$FTDOCK/build -in 1BRS.ftdock -b1 1 -b2 10
```

This command will create pdb files for the top 10 docking solutions (from 1 to 10) in the FTDock output file (pdb files will be named *Complex_1g.pdb*, *Complex_2g.pdb*, etc).

5.3. Visualization of docking solutions

You can use your favourite graphics viewer (VMD, PyMol, RasMol, etc.) to visualize the pdb files of the docking solutions.

5.3.1. Visualization with PyMol

Launch PyMol:

```
pymol
```

5.3.1.1. Visualize ZDOCK results

Compare the resulting docking conformations (`complex.1`, `complex.2`, etc.) with the crystallographic structure.

WARNING: The SYM data within the ZDOCK output is giving problems in PyMol. Therefore, before reading these output docking files with PyMol, we need to fix them by keeping only ATOM records:

```
cat complex.1 | grep ATOM > complex1.pdb
```

In PyMol, we can load the first docking solution from ZDOCK:

```
PyMOL> load complex1.pdb
```

WARNING: One problem with our first docking example (section 3) is that receptor and ligand files had the same chain ID ("A"), so in the ZDOCK output files, both chains were named with the same ID, and PyMol cannot distinguish them. One solution is to edit the docking file `complex1.pdb` to rename the second chain to "B", before loading it in PyMol. This can also be done within PyMol, following these commands:

1) Select atoms corresponding to the first chain as sorted in the original file (OXT atom does not count):

```
PyMOL> select chain1, rank 1-878
```

2) Select atoms corresponding to the second chain as sorted in the original file (OXT atom does not count):

```
PyMOL> select chain2, rank 879-1598
```

3) Rename chain ID of atoms in the second chain (to "B"):

```
PyMOL> alter chain2, chain='B'
```

We can display this docking solution (backbone representation), and colour receptor and ligand in white and red, respectively:

```
PyMOL> hide
PyMOL> show lines, name ca+c+n
```

(to display all atoms: `show lines`)

```
PyMOL> color white, ///A
PyMOL> color red, ///B
```

Now we can load the real complex and superimpose the receptor of the real complex onto that of the docking solution:

```
PyMOL> load 1BRS_AD.pdb
PyMOL> hide /1BRS_AD
PyMOL> show lines, /1BRS_AD and name ca+c+n
PyMOL> color green, /1BRS_AD
PyMOL> align /1BRS_AD//A, /complex1//A
```

Now that the receptors are superimposed, you can calculate the RMSD of the ligand C-alpha atoms with respect to the crystal structure. However, since the number of ligand atoms in x-ray and in docking model are different, we cannot use directly the command `rms_cur`. Before that, we need to align ligand atoms, without transforming the coordinates, and then, using this alignment, RMSD is calculated:

```
PyMOL> align /1BRS_AD//D//ca, /complex1//B//ca, cycles=0, transform=0, object=aln
PyMOL> rms_cur /1BRS_AD//D//ca & aln, /complex1//B//ca & aln, matchmaker=-1
```

5.3.1.2. Visualize FTDOCK results

To load the first docking solution from FTDOCK:

```
PyMOL> load Complex_1g.pdb
```

WARNING: In the first docking example (section 3), receptor and ligand files had the same chain ID ("A"), so in the FTDOCK output files, both chains were named with the same ID, and PyMol cannot distinguish them. One solution is to edit the docking file `Complex_1g.pdb` to rename the second chain to "B", before loading it in PyMol. This can also be done within PyMol, following these commands:

1) Select atoms corresponding to the first chain as sorted in the original file (OXT atom does not count):

```
PyMOL> select chain1, rank 1-878
```

2) Select atoms corresponding to the second chain as sorted in the original file (OXT atom does not count):

```
PyMOL> select chain2, rank 879-1598
```

3) Rename chain ID of atoms in the second chain (to "B"):

```
PyMOL> alter chain2, chain='B'
```

5.3.2. Visualization with ICM **(OPTIONAL)**

Here you will see some guidelines to use the ICM-Browser program (www.molsoft.com) to analyze the docking results and compare them with the known structure of the complex. ICM has powerful molecular graphics capabilities with a complete scripting language. You can create loops, make scripts to be run in background, create arrays, etc. You can read the online manual with a web browser:

```
www.molsoft.com/icm/
```

Launch the ICM program:

```
ssh -X $USER@localhost
icm
```

5.3.2.1. Visualize ZDOCK results

Compare the resulting docking conformations with the crystallographic structure.

In ICM, we can load the first docking solution by:

```
icm> read pdb “complex.1”
```

We can display this docking solution (backbone representation), and colour receptor and ligand in white and red, respectively:

```
icm> display a_//ca,c,n
```

(to display all atoms: `display a_//*`)

```
icm> color a_1 white
icm> color a_2 red
```

Now we can load the real complex and superimpose the receptor of the real complex onto that of the docking solution:

```
icm> read pdb "1BRS_AD.pdb"
icm> superimpose a_1.1//ca a_2.1//ca align
icm> display a_1,2//ca,c,n
icm> color a_1,2 grey
```

You can calculate the RMSD of the ligand C-alpha atoms with respect to the crystal structure:

```
icm> a = Srmsd(a_1.2//ca a_2.2//ca align)
icm> print a
```

5.3.2.2. Visualize FTDock results

Run similar commands to previous points, in order to visualize the pdb files created from FTDock docking.

WARNING:

One problem with our first docking example (section 3) is that receptor and ligand files had the same chain ID ("A"), so in the output files, both chains were named with the same ID, and ICM cannot distinguish them. ZDOCK already inserts a "TER" between chains, so ZDOCK files are fine, but FTDOCK does not insert a "TER" between chains and it adds extra lines that make ICM to have problems.

Therefore, before reading these FTDOCK output docking files with ICM, we need to fix them:

1) keep only ATOM records:

```
cat Complex_1g.pdb | grep ATOM > Complex_1.pdb
```

2) edit the new file "Complex_1.pdb" with a text editor and include a "TER " line between chains, like:

```
...
ATOM    877  NH1 ARG A 110      -6.709   8.054  10.050  1.00 14.54
ATOM    878  NH2 ARG A 110      -7.755   6.192  10.855  1.00 16.55
ATOM    879  OXT ARG A 110       0.151  10.166  10.282  1.00 14.66
TER
ATOM      1  N   LYS A   1     -26.540 -14.085  17.495  1.00 52.69
ATOM      2  CA  LYS A   1     -26.781 -14.791  16.200  1.00 51.62
ATOM      3  C   LYS A   1     -25.596 -14.625  15.243  1.00 49.92
...
```

WARNING: there should be at least one space after "TER " so that ICM can recognize this 4-character field.

You should repeat this process with all FTDock output files from this first example, so that they can be visualized with ICM.

General questions for discussion:

- How are the docking solutions stored in the ZDOCK and FTDOCK outputs? Where is the scoring? Can you find any differences in the scoring values of the two methods?

- How close are the highest-scoring FTDock and ZDOCK solutions to the x-ray structure?

- How many docking solutions with RMSD<10Å can be found in the top 10 solutions from each program?

- Will the combination of results from both programs improve the chances of a good prediction?

- From this analysis on a test case in which complex structure is known, have we learned any suggestion to optimally apply these docking programs in a real case situation?

REFERENCES

[1] Janin J, Henrick K, Moult J, Eyck LT, Sternberg MJE, Vajda S, Vakser I, Wodak SJ. **CAPRI: A Critical Assessment of PRedicted Interactions Proteins.** *Proteins.* 2003 Jul (1):2-9.

[2] Grosdidier S, Pons C, Solernou A, Fernández-Recio J. **Prediction and scoring of docking poses with pyDock.** *Proteins*. 2007 Dec 1;69(4):852-8.