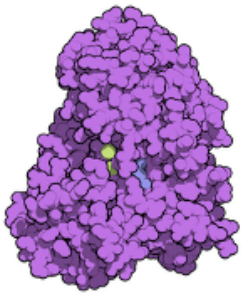

LightDock 2UUY example

1. Quick and dirty Protein-protein docking

The simplest way to perform a protein-protein docking in LightDock is to use default parameters and to only provide two PDB files for both receptor and ligand proteins.

1.1. Simplest example

You fill find in the examples a folder 2UUY with a PDB file for the receptor 2UUY_rec.pdb and the ligand 2UUY_lig.pdb.

In previous versions of LightDock, a setup step was not required. This has changed from **version 0.5.0** and now a simulation setup is required. Follow the next steps in order to perform your first protein-protein docking with LightDock:

1.1.1. Copying data

Create a directory and copy the sample data provided:

```
cd $LIGHTDOCK_HOME
cd examples
mkdir test
cd test
cp ../2UUY/2UUY*.pdb .
```

1.1.2. LightDock setup

Execute `lightdock_setup` script in order to prepare your LightDock simulation:

```
lightdock_setup 2UUY_rec.pdb 2UUY_lig.pdb 1 10
```

The previous command executes LightDock with the default parameters and only calculating 1 swarm containing 10 glowworms.

Here it is the output:

```
@> ProDy is configured: verbosity='info'
[lightdock_setup] INFO: Reading structure from 2UUY_rec.pdb PDB file...
[lightdock_setup] INFO: 1628 atoms, 223 residues read.
[lightdock_setup] INFO: Reading structure from 2UUY_lig.pdb PDB file...
[lightdock_setup] INFO: 415 atoms, 55 residues read.
[lightdock_setup] INFO: Calculating reference points for receptor 2UUY_rec.pdb...
[lightdock_setup] INFO: Done.
[lightdock_setup] INFO: Calculating reference points for ligand 2UUY_lig.pdb...
[lightdock_setup] INFO: Done.
[lightdock_setup] INFO: Saving processed structure to PDB file...
[lightdock_setup] INFO: Done.
[lightdock_setup] INFO: Saving processed structure to PDB file...
[lightdock_setup] INFO: Done.
[lightdock_setup] INFO: Calculating starting positions...
[lightdock_setup] INFO: Generated 1 positions files
[lightdock_setup] INFO: Done.
[lightdock_setup] INFO: Preparing environment
[lightdock_setup] INFO: Done.
[lightdock_setup] INFO: LightDock setup OK
```

A new file called `setup.json` has been generated with the simulation information .

If you execute `lightdock_setup` without arguments a little help is displayed:

```
usage: lightdock_setup [-h] [--seed_points STARTING_POINTS_SEED]
                       [-ft ftdock_file] [--noxt] [-anm] [--seed_anm ANM_SEED]
                       [-anm_rec ANM_REC] [-anm_lig ANM_LIG] [-rst restraints]
                       [-membrane]
                       receptor_pdb_file ligand_pdb_file swarms glowworms
lightdock_setup: error: too few arguments
```

1.1.3. LightDock run

Execute `lightdock` script in order to run your first LightDock simulation:

```
lightdock setup.json 10
```

You will see an output like this:

```
@> ProDy is configured: verbosity='info'
[lightdock] INFO: simulation parameters saved to ./lightdock.info
[lightdock_setup] INFO: Reading structure from 2UUY_rec.pdb PDB file...
[lightdock_setup] INFO: 1628 atoms, 223 residues read.
[lightdock_setup] INFO: Reading structure from 2UUY_lig.pdb PDB file...
[lightdock_setup] INFO: 415 atoms, 55 residues read.
[lightdock] INFO: Loading scoring function...
[lightdock] INFO: Using DFIRE scoring function
[lightdock] INFO: Done.
[kraken] WARNING: Number of cores has not been specified or is incorrect. Using av
ailable cores.
[kraken] INFO: Kraken has 4 tentacles (cpu cores)
[kraken] INFO: Tentacle ready with 1 tasks
[kraken] INFO: Tentacle ready with 0 tasks
[kraken] INFO: Tentacle ready with 0 tasks
[kraken] INFO: Tentacle ready with 0 tasks
[kraken] INFO: 1 ships ready to be smashed
[lightdock] INFO: Monster spotted
[kraken] INFO: Release the Kraken!
[kraken] INFO: folding tentacle Tentacle-2
[0] step 1
[kraken] INFO: folding tentacle Tentacle-4
[kraken] INFO: folding tentacle Tentacle-3
[0] step 2
[0] step 3
[0] step 4
[0] step 5
[0] step 6
[0] step 7
[0] step 8
[0] step 9
[0] step 10
[kraken] INFO: folding tentacle Tentacle-1
[kraken] INFO: 1 ships destroyed
[lightdock] INFO: Finished.
```

By default, LightDock makes use of the DFIRE scoring function. There is a warning on the number of CPU cores used. By default, LightDock will look for the total number of cores. If you want to specify a different number, use the flag `-c NUMBER_CORES`. **Note that MPI is also supported using the -mpi flag**.

For each of the swarms, there is a folder called `swarmX`. *In our example, we use only one swarm so there is only a folder* `swarm 0`. Inside, we can find the file containing the result of the simulation `gso_10.out`. See an example of this file from the *2UUY* example: gso_5.out.

In this file, every line corresponds to a glowworm agent in the algorithm:

```
#Coordinates  RecID  LigID  Luciferin  Neighbor's number  Vision Range  Scoring
(31.4171143,  1.8570079, -6.3956223, -0.1058407, -0.4849369,  0.5997430, -0.627648
2)    0    0  10.79432165  0 2.200   7.52191884
```

Finally, to generate the final docked PDB structures, we will use the script **lgd*generate*conformations.py**:

```
cd swarm_0
lgd_generate_conformations.py ../2UUY_rec.pdb ../2UUY_lig.pdb gso_10.out 10
```

Output is:

```
@> ProDy is configured: verbosity='info'
[generate_conformations] INFO: Reading ../lightdock_2UUY_rec.pdb receptor PDB file
...
[generate_conformations] INFO: 1628 atoms, 223 residues read.
[generate_conformations] INFO: Reading ../lightdock_2UUY_lig.pdb ligand PDB file..
.
[generate_conformations] INFO: 415 atoms, 55 residues read.
[generate_conformations] INFO: Read 10 coordinate lines
[generate_conformations] INFO: Generated 10 conformations
```

Inside the `swarm_0` folder 10 new PDB structures corresponding to the 10 glowworm agents used in the example have been generated.