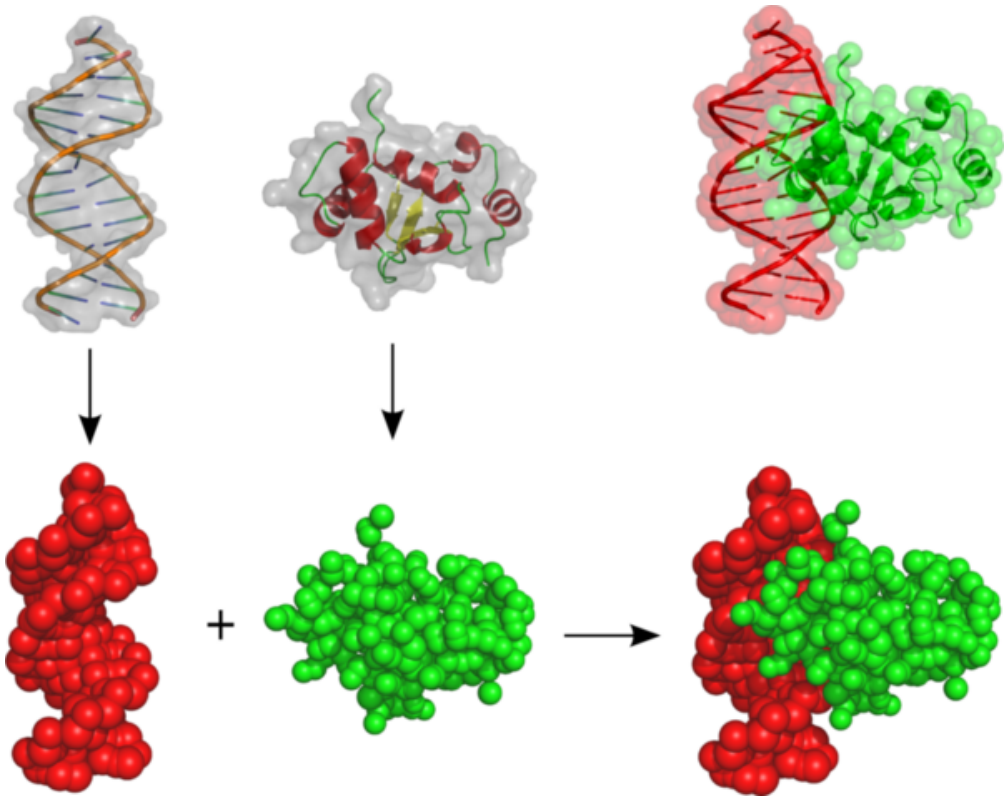
PTools tutorial

May 17, 2008



This tutorial presents the PTools library features and its docking application ATTRACT.

Contents

1 Set up, compilation and installation **3**
1.1 Set up . . . 3
1.2 Source download with subversion . . . 4
1.3 Compilation . . . 4
1.4 Final test and further documentation . . . 4

2 PTools library usages and capabilities **5**
2.1 Directly from C++ . . . 5
2.2 From Python through the C++ binding . . . 5
2.2.1 Rigidbody objects . . . 5
2.2.2 Selection objects . . . 6

3 Docking with PTools: ATTRACT **6**
3.1 Extraction of the docking partners and coarse grain reduction . . . 6
3.1.1 Partner preparation . . . 7
3.1.2 Protein coarse grain reduction . . . 7
3.1.3 DNA coarse grain reduction . . . 8
3.2 Docking simulation . . . 8
3.2.1 Ligand positions . . . 8
3.2.2 ATTRACT parameters . . . 9
3.2.3 Initial docking simulation . . . 9
3.3 Docking output analysis . . . 10

4 Misc. tips and tricks **10**

1 Set up, compilation and installation

1.1 Set up

This example installation has been performed on a Dell D420 laptop (Intel Core Duo 1.2 GHz, Debian Lenny, linux kernel 2.6.22-2-686).

The basic requirements are:

- Python 2.5 and its development library (Python2.5-dev)
- gcc (4.3)
- gfortran (4.3)

Scons (make substitute): `apt-get install scons`
(1.34.1-11 version installed)

Libboost-dev: `apt-get install libboost-dev`
(1.34.1-11 version installed)

Libboost Python: `apt-get install libboost-python1.34.1`
(1.34.1-11 version installed)

Libboost Python dev: `apt-get install libboost-python-dev`
(1.34.1-11 version installed)

Subversion: `apt-get install subversion subversion-tools`
(1.4.6dfsg1-2 version installed)

Gccxml: `apt-get install gccxml`
(0.9.0+cvs20071228-2 version installed)

Pyplusplus and Pygccxml: The homepage of `pyplusplus` and `pygccxml` projects is `http://www.language-binding.net/pyplusplus/pyplusplus.html`. From the download section [1], get the files `pygccxml-0.9.5.zip` and `Py++-0.9.5.zip`.

```
unzip pygccxml-0.9.5.zip
cd pygccxml-0.9.5/
python setup.py build
python setup.py install --prefix=$HOME/soft
```

```
unzip Py++-0.9.5.zip
cd Py++-0.9.5/
python setup.py build
python setup.py install --prefix=$HOME/soft
```

In your `$HOME/.bashrc` file, then add:

```
export PATH=PATH:$HOME/soft/bin/
export PYTHONPATH=$HOME/soft/lib/python2.4/site-packages/
```

In a Python shell (obtained with the `python` command), test the installation of PyPlus-Plus:

```
>>> import pyplusplus
```

[1] `https://sourceforge.net/project/showfiles.php?group_id=118209`

1.2 Source download with subversion

From the local `$HOME/soft/` directory, download the full PTools sources with subversion (use the *checkout* option):

```
svn co http://svn-lbt.ibpc.fr/svn/PTools/ptools ptools
```

- for the first use, press *Enter* and then indicate your login and password
- both login and password are stored (without encryption) in the `$HOME/.subversion` directory

PTools source updates are then obtained by:

```
svn update
```

from the `ptools/` or `ptools/trunk` directories.

1.3 Compilation

From the main directory (`trunk`) of the PTools project, create the Python/C++ interface:

```
python interface.py
```

Compile then the library:

```
scons
```

Note that `scons -j2` compiles with two processors in parallel.

1.4 Final test and further documentation

In the `Test` directory, one can test the compilation worked:

```
python unittest1.py
```

The expected output is:

```
.......
----------------------------------------------------------------------
Ran 7 tests in 0.813s

OK
```

Further document may be obtained from the Trac server `http://svn-lbt.ibpc.fr/PTools/wiki`. The server access is controled by the same login/password of the subversion server. The `Timeline` and `Browse Source` sections are usually very usefull.

A `README` file is also available on line [2] or locally [3].

[2] `http://svn-lbt.ibpc.fr/PTools/browser/ptools/trunk/Tutorial/README`

[3] `$HOME/soft/ptools/trunk/Tutorial/README`

2 PTools library usages and capabilities

2.1 Directly from C++

2.2 From Python through the C++ binding

Explain here the different level of objects between rigidbody, selection, atom and points

From the Python interpreter or in a Python script, first load the PTools library:

```
from ptools import *
```

2.2.1 Rigidbody objects

Load PDB file into a rigidbody object.

```
pdb = Rigidbody("1BTA.pdb")
```

Number of atoms.

```
pdb.Size()
```

Maximum distance from geometric center in Å.

```
pdb.Radius()
```

Radius of gyration in Å.

```
pdb.RadiusGyration()
```

Structure translation. First create a translation vector as a Coord3D object (for instance 5, 0, 1):

```
trans = Coord3D(5, 0, 1)
```

Then, apply the translation vector:

```
pdb.Translate(trans)
```

Center structure to origin.

```
pdb.CenterToOrigin()
```

Save structure as PDB file.

```
WritePDB(pdb, "1BTA_centered.pdb")
```

2.2.2 Selection objects

Selection of CA atoms.

```
sel_ca = pdb.CA()
```

Selection of backbone atoms.

```
sel_bkbn = pdb.Backbone()
```

Selection by chain.

```
sel_chainA = pdb.SelectChainId("A")
sel_chainB = pdb.SelectChainId("B")
```

Selection of a range of residues.

```
sel_res = pdb.SelectResRange(10, 20)
```

Selection number of atoms.

```
sel_res.Size()
```

Selection reunion.

```
sel_chainAB = sel_chainA | sel_chainB
```

or directly

```
sel_chainAB = pdb.SelectChainId("A") | pdb.SelectChainId("B")
```

Selection to rigidbody conversion.

```
ca_trace = sel_ca.CreateRigid()
```

3 Docking with PTools: ATTRACT

This part is illustrated by the docking of the protein and DNA found in the 1DNJ complex.

3.1 Extraction of the docking partners and coarse grain reduction

Before docking, one has to separate both partners. This is possible with visualisation software such as Pymol [4] or VMD [5], and also directly with the PTools library.

[4] http://pymol.sourceforge.net/
[5] http://www.ks.uiuc.edu/Research/vmd/

3.1.1 Partner preparation

Within the Python interpreter, first load the PTools library:

```
from ptools import *
```

Read the PDB file 2NJ.pdb:

```
pdb=Rigidbody("2DNJ.pdb")
```

The chain selection allows the separation between the protein (chain A) and the DNA (chains B and C).

```
selectA=pdb.SelectChainId("A")
selectB=pdb.SelectChainId("B")
selectC=pdb.SelectChainId("C")
```

Reunion of both DNA strands:

```
selectBC=selectB | selectC
```

Creation of the protein as a rigid body and file saving as a PDB:

```
prot=selectA.CreateRigid()
WritePDB(prot,"2DNJ_prot.pdb")
```

Creation of the DNA as a rigid body and file saving as a PDB:

```
DNA=selectBC.CreateRigid()
WritePDB(DNA,"2DNJ_dna.pdb")
```

Once both partners are isolated in separe files, close the Python interpreter.

To check DNA atom names are compatible with the one used is the reduced model, use the `parse.pl` script:

```
parse.pl 2DNJ_dna.pdb >2DNJ_dnaf.pdb
```

3.1.2 Protein coarse grain reduction

```
reduce.py 2DNJ_prot.pdb 2DNJ_prot.cg.z.pdb
```

Parameters:

- an all-atom protein pdb file (`2DNJ_prot.pdb`)
- a reduced output pdb file (`2DNJ_prot.cg.z.pdb`)

Visualise both to check that the reduction worked properly.

3.1.3 DNA coarse grain reduction

DNA is reduced using two scripts: `reduceDna.py` and `convertPdb2Zacharias.py`. The first one perform the proper reduction.

```
reduceDna.py atom2cgDna.dat 2DNJ_dnaf.pdb 2DNJ_dna.cg.pdb
```

Parameters are the following:

- the correspondance between atoms and beads (`atom2cgDna.dat`)
- an input all-atom pdb file (`2DNJ_dnaf.pdb`)
- an output coarse grain pdb file (`2DNJ_dna.cg.pdb`)

The second program converts the output DNA reduced pdb file into a pdb-like file compatible with ATTRACT. Its usage is:

```
convertPdb2Zacharias.py ffParamDna.dat 2DNJ_dna.cg.pdb 2DNJ_dna.cg.z.pdb
```

with the parameters:

- the parameters of the coarse grain force field (bead type and charge) (`ffParamDna.dat`)
- an input coarse grain pdb file (`2DNJ_dna.cg.pdb`)
- an output coarse grain pdb file compatible with ATTRACT (`2DNJ_dna.cg.z.pdb`)

As for protein reduction, a visual inspection with any visualisation software is preferable.

3.2 Docking simulation

3.2.1 Ligand positions

The file `translat.dat` contains all translational initial positions of the ligand used for the systematic docking. To produce this file, the largest dimension of the ligand is needed and obtained with the Python script `rayon.py`:

```
rayon.py 2DNJ_dna.cg.z.pdb
```

In the present case, the largest dimension of the ligand is 15.88 Å. The translation distance around the receptor has then to be of least of 15.88 Å, 17 Å should be fine here.

Starting position (in translation) for the docking are then obtained by:

```
translate 2DNJ_prot.cg.z.pdb 17 > translat.dat
```

The `translate` program creates regurlarly spread points at 17 Å from the surface of the receptor.

The visualisation of the starting points may be viewed with any visualisation software by renaming `translat.dat` in `translat.pdb`.

3.2.2 ATTRACT parameters

ATTRACT parameters are found in the file `attract.inp`. A typical configuration file is:

```
 6 0 0
-34.32940 38.75490 -3.66956 0.00050
 30  2 1 1 0 0 0 0 1 2000.00
 30  2 1 1 0 0 0 0 1 1000.00
 50  2 1 1 0 0 0 0 0  500.00
 50  2 1 1 0 0 0 0 0   50.00
 100 2 1 1 0 0 0 0 0   50.00
 500 2 1 1 0 0 0 0 0   50.00
```

The first line indicates the number of minimisations performed by ATTRACT for each starting position. The last six lines are the caracteristics of each minimisation. The first column is the number of steps before the minimisation stops. The last column is the square of the cutoff distance for the calculation of the interaction energy between both partners.

Remarques: Columns with zeros or ones should not be modfied, as well as the second line.

3.2.3 Initial docking simulation

A docking simulation with ATTRACT requires:

- a receptor (fixed partner) file (`2DNJ_prot.cg.z.pdb`)
- a ligand (mobile partner) file (`2DNJ_dna.cg.z.pdb`)
- the ATTRACT program (here for protein–DNA docking, `Attract_dna.py`)
- the coarse grain parameters (`aminon.dna.par`)
- rotations performed for each translational starting point (`rotation.dat`)
- translational starting points (`translat.dat`)
- docking parameters (`attract.inp`)

ATTRACT can be used with different options:

- -s, performs one single serie of minimisations with the ligand in its initial position.
- -ref=, provides a ligand pdb file as a reference. After each docking, the RMSD is calculated between this reference structure and the simulated ligand.

A single ATTRACT simulation may thus be obtained by:

```
Attract_dna.py 2DNJ_prot.cg.z.pdb 2DNJ_dna.cg.z.pdb
-s -ref=2DNJ_dna.cg.z.pdb > 2DNJ_single.att
```

The first pdb file provided must be the receptor file (and the second the ligand).
For a full systematic docking in the translational and rotational space:

```
Attract_dna.py 2DNJ_prot.cg.z.pdb 2DNJ_dna.cg.z.pdb
-ref=2DNJ_dna.cg.z.pdb > 2DNJ_dock.att
```

The output file `2DNJ_dock.att` contains all informations on the docking simulation.

3.3 Docking output analysis

Any simulated ligand structure can be extracted with the script `Extract.py`:

```
Extract.py 2DNJ_dock.att 2DNJ_dna.cg.z.pdb 101 153 > 2DNJ_dock.pdb
```

with the parameters:

- an ouput of the docking simulation (`2DNJ_dock.att`)
- the inital ligand file (`2DNJ_dna.cg.z.pdb`)
- a translation index (`101`)
- a rotation index (`153`)
- an output ligand structure file (`2DNJ_dock.pdb`)

4 Misc. tips and tricks