System preparation

A pyMDMix project can contain several systems which are macromolecules to be studied and ready to be simulated in vacuo. A Replica, is a system for which the solvent has been added (a solvated system) and all MD inputs have been writen following particular simulation conditions. There are two ways for adding a system to a project:

1. Manually create an amber object file with the 'unsolvated' system. That is to prepare the system as if it was going to be simulated in vacuo.
2. Prepare a PDB file with correct protonation states and conforming names to Amber standards (e.g. HIE for epsilon protonated histidines, ASP for charged aspartic acid, CYX for disulfide bridging cysteines, etc.).

Creating a system with an amber object file

This is the first step of preparation for a pyMDMix run and is the preferred way (creation of systems from PDB files is discussed below). The main input file for pyMDMix is an Amber Object File (OFF file).

Here you will find a basic tutorial on how to prepare systems for simulation using Amber. In the tutorial, they follow up until simulation stages but the user of pyMDMix should stop once ***saveOff*** action is done in ***tLeap/xLeap*** and an Object File is generated. In the tutorial the object file is named ***1cgh-no-inhib.lib***.

The user should prepare the system as if it was going to be simulated in vacuo. It means the system should be completely parameterized using the chosen forcefield. Correctly protonated, capped, with flipped residues (if needed), and disulfide bridges connected. **Do not add solvent nor neutralize** with counter-ions (this is automatically done later by the program).

IMPORTANT! If there are special residues, ligands, ions or other custom parameterized residues in the system, add the corresponding parameters to the saved Object File (OFF) as well.

Example

We are interested in simulating a system which contains a NADH cofactor. We have prepared a PDB with the protonated protein and the cofactor inside it (here file `1p44_protonated.pdb`). Parameters for NADH cofactor can be downloaded from AMBER Parameter Database (here files named `nadh.prep` and `nadh.frcmod`). To build the object file of the system, we could issue the following commands in **tLeap**:

```
> NADH = loadAmberPrep nadh.prep
> NADHparams = loadAmberParams nadh.frcmod
> system = loadPdb 1p44_protonated.pdb
> check system # no parameters error expected
> saveOff system inha.off
> saveOff NADH inha.off
> saveOff NADHparams inha.off
```

Before finishing, it is recommended to try and save the Amber topology and coordinate files for the recently created OFF file. This step should not fail. Otherwise, there are parameterization errors that should be manually fixed before proceeding:

```
> loadOff inha.off
> saveAmberParm system inha.top inha.crd
```

Check last section on this site to learn how to use this OFF file generated to create a system in pyMDMix.

Creating a system with a PDB

This option is still experimental but it will be possible to create a system using a clean PDB file. By clean, I mean all residue names conform to Amber standards, protonation states are correctly set, there are no residues missing and non-standard residues have been parameterized.

Basically this way of creating the system will save you the process described in previous section and will try to do it automatically. It will only work if the PDB is well prepared. Check next section on how to create the system this way.

Adding the system to pyMDMix

Once the OFF (or PDB) file is ready, we have to tell the program we want to work with this system. The way to do it is by means of a **System Configuration File (SCF)**. In this file, we will give the path to the input file and tell the program about force field used for parameterization or non-standard residue names that might be present in the system. The config file template has this shape:

```
# pyMDMix System Configuration file
# WARNING: All empty options, should be removed from the file or commented
[SYSTEM]
# Identify the system
NAME =
# One of these two options should be given:
# Path to amber object file containing the system unit (without solvation). MANDATORY (unless PDB is give
# Path to PDB file that should be prepared as an object file (experimental). MANDATORY (unless OFF is giv
OFF =
PDB =
# Unit name containing the system inside the object file (default: first unit found)
#UNAME =
# Comma separated list of non-standard residues that should be considered part of the system and not the
# Used in automask detection and solute-solvent identification (default: empty)
#EXTRARES =
# Forcefields or Forcefield modification files (frcmod) we should consider when parameterizing the solvat
# If the object file was prepared using leaprc.ff12SB, give it here. If the system contains non-standard
# that you have parameterized, give the frcmod files here (and include the units inside the object file)
# By default, leaprc.ff99SB is considered
#EXTRAFF =
```

For the example above, this would be the configuration file needed:

```
# File saved as mysystem.cfg
[SYSTEM]
NAME = SystemA
OFF = inha.off
UNAME = system
EXTRARES = NADH
```

Final step is to add the system to an existing project. Move to the project folder and type:

```
# Move to an existing pymdmix project folder
> mdmix add system -f mysystem.cfg
```

If we look at the project information we should find a system named `SystemA`.