Gromacs Py Documentation

Release 2.0.3-rc

Samuel Murail

Jul 05, 2022

USER DOCUMENTATION:

Gromacs_py is a Python library allowing a simplified use of the Gromacs MD simulation software. **Gromacs_py** can build a system topologie based on a pdb file, create the simulation system (pbc box, adding water and ions) and run minimisation, equilibration and production runs. One of the main objective of the **Gromacs_py** wrapper is to automatize routine operations for MD simulation of multiple systems.

Gromacs_py is under active development using continuous integration with Travis Cl.

- **Online Documentation:**
  https://gromacs-py.readthedocs.io
- **Source code repository:**
  https://github.com/samuelmurail/gromacs_py

CHAPTER

ONE

QUICK INSTALL

The latest release can be installed via *pip* or *conda*.

1.1 Conda

If you don't need a GPU compiled version of Gromacs you can use directly the **Gromacs_py** conda package to install both Gromacs software and **Gromacs_py** library:

```
conda install -c bioconda gromacs_py
```

1.2 Pip (Deprecated)

If the folowing softwares/modules are installed then you need to install the **Gromacs_py** library using pypi :

```
pip install gromacs_py
```

- Gromacs (version >= 5.1)
- Ambertools
- Rdkit
- Acpype

and add the software path Gromacs in the environmnent variable `$PATH`, *eg.* for gromacs:

```
# Add gromacs 'gmx' path:
export PATH='*path_to_gromacs*/bin/':$PATH
```

CHAPTER

TWO

AUTHORS

- Samuel Murail, Associate Professor - Université Paris Diderot, CMPLI.

See also the list of contributors who participated in this project.

CHAPTER

THREE

LICENSE

This project is licensed under the GNU General Public License v2.0 - see the `LICENSE` file for details.

CHAPTER

FOUR

MAIN FEATURES:

- **Python Scriptable simulation:**
  - Topologie creation
  - Solvation
  - Ion insertion
  - Energy minimisation
  - Equilibration with different position restraints
  - Production
- **Topologie manipulation starting from a raw `PDB`:**
  - Amino acid protonation and pKa calculation using apbs/pdb2pqr
  - Position constraints file `.itp` creation
  - Cyclic peptide topologie
  - Cystein bond topologie modification
  - ligand topologie using *ambertools* and *acpype*
- **Advanced simulation tools:**
  - Monitor a simulation while running
  - Free Energy calculations
  - Interrupt a simulation if a criterion is met (Not implemented yet)

CHAPTER

FIVE

COMPATIBILITY

- **Supported Gromacs versions:**
  - 2020
  - 2019*
  - 2018*
  - 2017
  - 2016
  - 5.1
  - 5.0
- **Supported Python versions:**
  - 3.8*
  - 3.7*
  - 3.6*
- **Supported OS:**
  - osx*
  - linux*

* tested after each code submission.

CHAPTER

SIX

INDICES AND TABLES

- genindex
- modindex
- search

CHAPTER

SEVEN

INSTALLATION

7.1 1. Get sources from the GithubRepo

The sources for Gromacs_py can be downloaded from the GithubRepo.

You can either clone the public repository:

```
$ git clone git://github.com/samuelmurail/gromacs_py
```

Or download the tarball:

```
$ curl -OJL https://github.com/samuelmurail/gromacs_py/tarball/master
```

Once you have a copy of the source, switch to the `gromacs_py` directory.

```
$ cd gromacs_py
```

7.2 2. Create Conda Environment

You need to create a conda environment to be able to use:

- Gromacs
- Rdkit Used for ligand parametrization, convert SMILE to pdb.
- Antechamber Amber tools for ligand parametrization.
- Acpype a python tool to use antechamber.
- Apbs Pdb2pqr Protein protonation calculation.

Use *conda en create* to create it using the `.conda.yml` file. You can overide the environmnent name using the option `--name YOUR_NAME`.

```
$ conda env create -f .conda.yml
```

If you plan to use `gromacs_py` in jupyter notebook, you should try the `jupyter` version:

```
$ conda env create -f .conda_jupyter.yml
```

This will create an environmnet called `gromacs_py` (or the name you defined). You will then, need to activate the environmnent:

```
$ conda activate gromacs_py
```

Note: If you want to install yourself Gromacs to be able to use the GPU acceleration, you can use the `.conda_no_gromacs.yml` or `.conda_jupyter_no_gromacs.yml`:

```
$ conda env create -f .conda_no_gromacs.yml
$ conda activate gromacs_py
```

7.3 3. Install gromacs_py

Once you have a copy of the source and have create a conda encironment, you can install it with:

```
$ python setup.py install
```

7.4 4. Test Installation

To test the installation, simply use `pytest`:

```
$ pytest
============================= test session starts ==============================
platform linux -- Python 3.8.2, pytest-5.4.2, py-1.9.0, pluggy-0.13.1
rootdir: /home/murail/Documents/Code/gromacs_py, inifile: pytest.ini
plugins: cov-2.10.1
collected 30 items

gromacs_py/gmx.py .............                                          [ 43%]
gromacs_py/test/test_FreeEner.py ......                                  [ 63%]
gromacs_py/test/test_GmxSys.py ..                                        [ 70%]
gromacs_py/tools/ambertools.py ....                                      [ 83%]
gromacs_py/tools/monitor.py .....                                        [100%]

====================== 30 passed in 236.83s (0:03:56) ======================
```

7.5 Conda installation

If you don't need a GPU compiled version of Gromacs you can use directly the **Gromacs_py** conda package to install both Gromacs software and **Gromacs_py** library:

```
conda install -c bioconda gromacs_py
```

7.5.1 Pypi (Deprecated)

If gromacs (version >= 5.1) is already install, then install you need to install the *gromacs_py* library, and add the gromacs *gmx* command in the environmnent variable *$PATH*:

```
pip install gromacs_py

# Add gromacs 'gmx' path:
export PATH='*path_to_gromacs*/bin/':$PATH
```

7.6 Without Conda

Get the gromacs_py library from github.

```
git clone https://github.com/samuelmurail/gromacs_py.git
./setup.py install --user

# Add gromacs 'gmx' path:
export PATH='*path_to_gromacs*/bin/':$PATH
```

7.6.1 Prerequisites

1. **python 3 libraries installed when you launch the pip command:**
    - numpy
    - scipy
    - pandas
    - matplotlib
    - Sphinx and sphinx-argparse (only for building documentation)
    - Os_Command_py
    - PDB_Manip_py
    - PDB2PQR using the package pdb2pqr_htmd_propka30 a python 3 version developped by tonigi and adapted to use successfully propka3.0.
2. Gromacs

Get source code from gromacs website and follow the following command for a quick and dirty install (for more details see gromacs 2019 install guide)

In my case I add to change few options to `cmake`:

- `-DCMAKE_C_COMPILER=gcc-8`, as gcc versions later than 6 are not supported.
- `-DGMX_GPU=on` to use GPU acceleration
- `-DCMAKE_INSTALL_PREFIX=../../local-gromacs-2019.2/` to install gromacs in a non-standard location

```
# Specify the version:
version="2021.5"
# To modify:
```

(continues on next page)

(continued from previous page)

```
dir_install="/home/murail/Documents/Software/local-gromacs-${version}"

wget https://ftp.gromacs.org/gromacs/gromacs-${version}.tar.gz
tar -xvf gromacs-${version}.tar.gz
cd gromacs-${version}
mkdir build
cd build
# In my case I needed to define ggc-8 because gromacs doesn't accept superior versions
cmake .. -DGMX_BUILD_OWN_FFTW=ON -DREGRESSIONTEST_DOWNLOAD=ON -DGMX_GPU=CUDA -DCMAKE_
↪INSTALL_PREFIX=${dir_install} -DCMAKE_C_COMPILER=gcc-8
make
make check
make install
source ${dir_install}/bin/GMXRC
echo "export PATH=${dir_install}/bin/:\$PATH" >> ~/.bashrc
```

3. Ambertools

The easiest way is to use the conda package. However is can also be installed from source.

CHAPTER

EIGHT

BASIC USAGE, SMALL PROTEIN

8.1 Gromacs_py basic example

Here is an example of a short simulation of the SH3 domain of phospholipase C$:nbsphinx-math:`gamma`$1. Five successive steps are used:

1. Topologie creation using `GmxSys.prepare_top()`.
2. Minimisation of the structure using `GmxSys.em_2_steps()`.
3. Solvation of the system using `GmxSys.solvate_add_ions()`.
4. Equilibration of the system using `GmxSys.em_equi_three_step_iter_error()`.
5. Production run using `GmxSys.production()`.

8.1.1 Import

[1]:
```
import sys
import os
import urllib.request
import pandas as pd
import matplotlib
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
```

- To use `gromacs_py` in a project:

[2]:
```
from gromacs_py import gmx
```

8.2 Simulation setup

- Define a few variables for you simulation, like:
  - simulation output folders
  - ionic concentration
  - number of minimisation steps
  - equilibration and production time

[3]:
```
DATA_OUT = 'data_sim'
PDB_ID = '1Y0M'

# System Setup
vsite='none'
ion_C = 0.15
sys_top_folder = os.path.join(DATA_OUT, 'sys_top')

# Energy Minimisation
em_folder = os.path.join(DATA_OUT, 'em')
em_sys_folder = os.path.join(DATA_OUT, 'sys_em')
em_step_number = 5000

# Equillibration
equi_folder = os.path.join(DATA_OUT, 'sys_equi')
HA_time = 0.5
CA_time = 1.0
CA_LOW_time = 2.0

dt_HA = 0.001
dt = 0.002

HA_step = 1000 * HA_time / dt_HA
CA_step = 1000 * CA_time / dt
CA_LOW_step = 1000 * CA_LOW_time / dt

# Production
prod_folder = os.path.join(DATA_OUT, 'sys_prod')
prod_time = 10.0

prod_step = 1000 * prod_time / dt
```

8.3 Get PDB file from the rcsb.org website

[4]:
```
os.makedirs(DATA_OUT, exist_ok = True)

raw_pdb = urllib.request.urlretrieve('http://files.rcsb.org/download/{}.pdb'.format(PDB_
↪ID),
                                     '{}/{}.pdb'.format(DATA_OUT, PDB_ID))
```

8.3.1 Create the GmxSys object

[5]:
```
md_sys = gmx.GmxSys(name=PDB_ID, coor_file=raw_pdb[0])
md_sys.display()
```

```
name         : 1Y0M
coor_file    : data_sim/1Y0M.pdb
nt           : 0
```

(continues on next page)

(continued from previous page)

```
ntmpi        : 0
sys_history  : 0
```

8.4 Create topology:

Topologie creation involves: - protonation calculation using `pdb2pqr` and `propka` - topologie creation using `pdb2gmx`

[6]:
```
md_sys.prepare_top(out_folder=os.path.join(DATA_OUT, 'prot_top'), vsite=vsite, ff=
↪'amber99sb-ildn')
md_sys.create_box(dist=1.0, box_type="dodecahedron", check_file_out=True)
```

8.4.1 3D coordinates vizualisation using `nglview`

Use the `view_coor()` function of the `GmxSys` object to vizualise the protein coordinates.

Note that `nglview` library need to be installed :

```
conda install nglview
```

You'll probably need to restart you notebook after installation to enable `nglview` widget appearance.

[7]:
```
view = md_sys.view_coor()
view.add_representation(repr_type='licorice', selection='protein')
view
```

```
NGLWidget()
```

[9]:
```
# Unecessary, only need to nglview online:
IFrame(src='../_static/1Y0M.html', width=800, height=300)
```

[9]:
```
<IPython.lib.display.IFrame at 0x7f36996b4580>
```

8.5 Energy minimisation

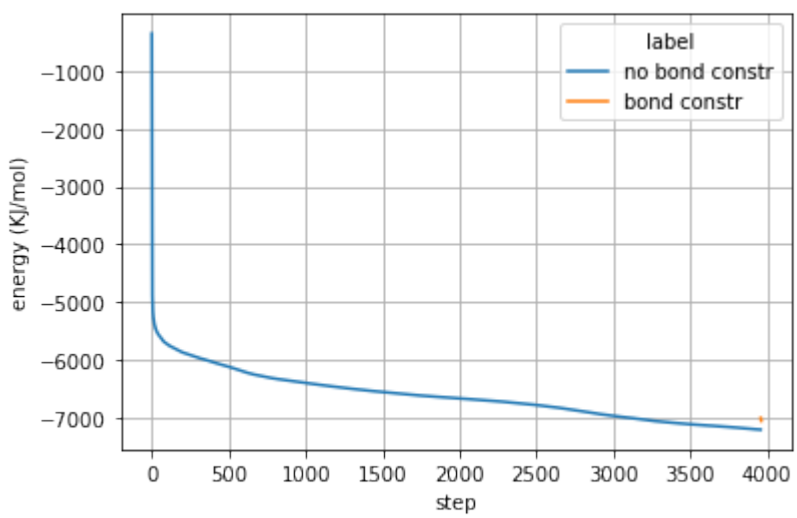
[12]:
```
md_sys.em_2_steps(out_folder=em_folder,
                  no_constr_nsteps=em_step_number,
                  constr_nsteps=em_step_number,
                  posres="",
                  create_box_flag=False,
                  emtol=0.1, nstxout=100)
```

8.6 Plot energy:

[30]:
```
ener_pd_1 = md_sys.sys_history[-1].get_ener(selection_list=['Potential'])
ener_pd_2 = md_sys.get_ener(selection_list=['Potential'])

ener_pd_1['label'] = 'no bond constr'
ener_pd_2['label'] = 'bond constr'

ener_pd = pd.concat([ener_pd_1, ener_pd_2])

ener_pd['Time (ps)'] = np.arange(len(ener_pd))
```

```
gmx energy -f data_sim/em/Init_em_1Y0M.edr -o tmp_edr.xvg
gmx energy -f data_sim/em/1Y0M.edr -o tmp_edr.xvg
```

[31]:
```
ax = sns.lineplot(x="Time (ps)", y="Potential",
                  hue="label",
                  data=ener_pd)
ax.set_xlabel('step')
ax.set_ylabel('energy (KJ/mol)')
plt.grid()
```



8.6.1 3D vizualisation using `nglview`

Not much append in the second minimisation, we can have a look at the first one using md_sys.sys_history[-1], which is considered as a `GmxSys` object.

Use the `coor_traj` atribute of the `GmxSys` object to vizualise the trajectory. Note that that the `simpletraj` library is a dependenie. To install `simpletraj` use:

```
pip install simpletraj
```

- first you should make molecule whole using `convert_trj()` function.

[14]:
```
md_sys.sys_history[-1].convert_trj()
```

[75]:
```
view = md_sys.sys_history[-1].view_traj()
view.add_representation(repr_type='licorice', selection='protein')
view.center()
view
```

```
NGLWidget()
```

[77]:
```
# Unecessary, only need to nglview online:
IFrame(src='../_static/1Y0M_em_traj.html', width=800, height=300)
```

[77]:
```
<IPython.lib.display.IFrame at 0x7f3685438fd0>
```

8.7 Solvation (water and Na^+Cl^-)

[18]:
```
md_sys.solvate_add_ions(out_folder=sys_top_folder,
                        ion_C=ion_C)
md_sys.display()
```

```
name         : 1Y0M
sim_name     : 1Y0M
coor_file    : data_sim/sys_top/1Y0M_water_ion.gro
top_file     : data_sim/sys_top/1Y0M_water_ion.top
tpr          : data_sim/em/1Y0M.tpr
mdp          : data_sim/em/1Y0M.mdp
xtc          : data_sim/em/1Y0M.trr
edr          : data_sim/em/1Y0M.edr
log          : data_sim/em/1Y0M.log
nt           : 0
ntmpi        : 0
sys_history  : 4
```

8.8 System minimisation and equilibration

[19]:
```
md_sys.em_equi_three_step_iter_error(out_folder=equi_folder,
                                     no_constr_nsteps=em_step_number,
                                     constr_nsteps=em_step_number,
                                     nsteps_HA=HA_step,
                                     nsteps_CA=CA_step,
                                     nsteps_CA_LOW=CA_LOW_step,
                                     dt=dt, dt_HA=dt_HA,
                                     vsite=vsite, maxwarn=1)
```

```
gmx mdrun -s equi_HA_1Y0M.tpr -deffnm equi_HA_1Y0M -nt 0 -ntmpi 0 -nsteps -2 -
↪nocopyright -append -cpi equi_HA_1Y0M.cpt
```

```
  0%|          | 0/500000 [00:00<?, ?it/s]
```

```
gmx grompp -f equi_CA_1Y0M.mdp -c ../00_equi_HA/equi_HA_1Y0M.gro -r ../../sys_em/1Y0M_
↪compact.pdb -p ../../../sys_top/1Y0M_water_ion.top -po out_equi_CA_1Y0M.mdp -o equi_CA_
↪1Y0M.tpr -maxwarn 1
gmx mdrun -s equi_CA_1Y0M.tpr -deffnm equi_CA_1Y0M -nt 0 -ntmpi 0 -nsteps -2 -nocopyright
```

```
  0%|          | 0/500000 [00:00<?, ?it/s]
```

```
gmx grompp -f equi_CA_LOW_1Y0M.mdp -c ../01_equi_CA/equi_CA_1Y0M.gro -r ../../sys_em/
↪1Y0M_compact.pdb -p ../../../sys_top/1Y0M_water_ion.top -po out_equi_CA_LOW_1Y0M.mdp -
↪o equi_CA_LOW_1Y0M.tpr -maxwarn 1
gmx mdrun -s equi_CA_LOW_1Y0M.tpr -deffnm equi_CA_LOW_1Y0M -nt 0 -ntmpi 0 -nsteps -2 -
↪nocopyright
```

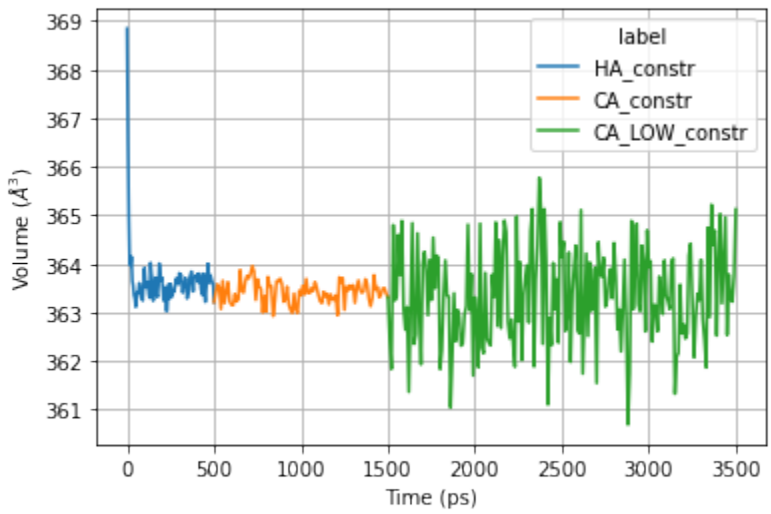
```
  0%|          | 0/1000000 [00:00<?, ?it/s]
```

8.8.1 Plot temperature

[35]:
```
ener_pd_1 = md_sys.sys_history[-2].get_ener(selection_list=['Volume'])
ener_pd_2 = md_sys.sys_history[-1].get_ener(selection_list=['Volume'])
ener_pd_3 = md_sys.get_ener(selection_list=['Volume'])

ener_pd_1['label'] = 'HA_constr'
ener_pd_2['label'] = 'CA_constr'
ener_pd_2['Time (ps)'] = ener_pd_2['Time (ps)'] + ener_pd_1['Time (ps)'].max()
ener_pd_3['label'] = 'CA_LOW_constr'
ener_pd_3['Time (ps)'] = ener_pd_3['Time (ps)'] + ener_pd_2['Time (ps)'].max()

ener_pd = pd.concat([ener_pd_1, ener_pd_2, ener_pd_3])
```

```
gmx energy -f data_sim/sys_equi/sys_equi/00_equi_HA/equi_HA_1Y0M.edr -o tmp_edr.xvg
gmx energy -f data_sim/sys_equi/sys_equi/01_equi_CA/equi_CA_1Y0M.edr -o tmp_edr.xvg
gmx energy -f data_sim/sys_equi/sys_equi/02_equi_CA_LOW/equi_CA_LOW_1Y0M.edr -o tmp_edr.
↪xvg
```

[36]:
```
ax = sns.lineplot(x="Time (ps)", y="Volume",
                  hue="label",
                  data=ener_pd)

ax.set_ylabel('Volume ($Å^3$)')
plt.grid()
```



8.8.2 Plot RMSD

[42]:
```
# Define reference structure for RMSD calculation
ref_sys =  md_sys.sys_history[1]

rmsd_pd_1 = md_sys.sys_history[-2].get_rmsd(['C-alpha', 'Protein'], ref_sys=ref_sys)
rmsd_pd_2 = md_sys.sys_history[-1].get_rmsd(['C-alpha', 'Protein'], ref_sys=ref_sys)
rmsd_pd_3 = md_sys.get_rmsd(['C-alpha', 'Protein'], ref_sys=ref_sys)


rmsd_pd_1['label'] = 'HA_constr'
rmsd_pd_2['label'] = 'CA_constr'
rmsd_pd_2['time'] = rmsd_pd_2['time'] + rmsd_pd_1['time'].max()
rmsd_pd_3['label'] = 'CA_LOW_constr'
rmsd_pd_3['time'] = rmsd_pd_3['time'] + rmsd_pd_2['time'].max()

rmsd_pd = pd.concat([rmsd_pd_1, rmsd_pd_2, rmsd_pd_3])
```

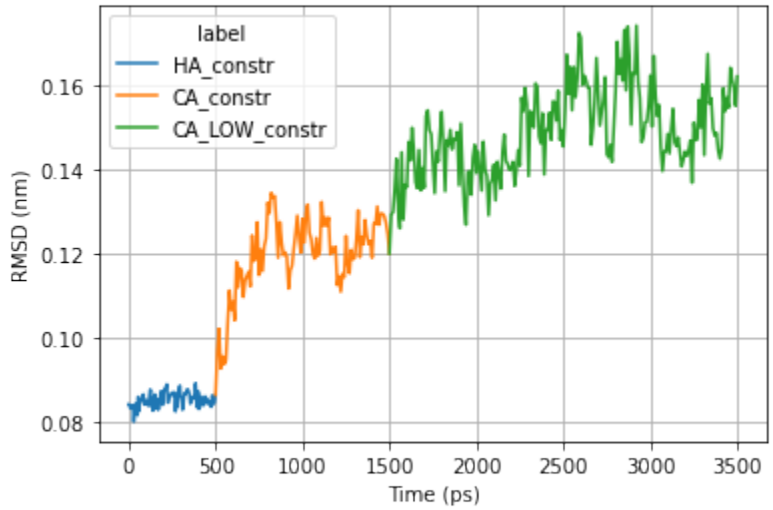
```
gmx rms -s data_sim/em/Init_em_1Y0M.tpr -f data_sim/sys_equi/sys_equi/00_equi_HA/equi_HA_
↪1Y0M.xtc -n data_sim/sys_equi/sys_equi/00_equi_HA/equi_HA_1Y0M.ndx -o tmp_rmsd.xvg -
↪fit rot+trans -ng 1 -pbc no
gmx rms -s data_sim/em/Init_em_1Y0M.tpr -f data_sim/sys_equi/sys_equi/01_equi_CA/equi_CA_
↪1Y0M.xtc -n data_sim/sys_equi/sys_equi/01_equi_CA/equi_CA_1Y0M.ndx -o tmp_rmsd.xvg -
↪fit rot+trans -ng 1 -pbc no
gmx rms -s data_sim/em/Init_em_1Y0M.tpr -f data_sim/sys_equi/sys_equi/02_equi_CA_LOW/
↪equi_CA_LOW_1Y0M.xtc -n data_sim/sys_equi/sys_equi/02_equi_CA_LOW/equi_CA_LOW_1Y0M.ndx␣
↪-o tmp_rmsd.xvg -fit rot+trans -ng 1 -pbc no
```

[43]:
```
ax = sns.lineplot(x="time", y="Protein",
                  hue="label",
                  data=rmsd_pd)

ax.set_ylabel('RMSD (nm)')
```

(continues on next page)

(continued from previous page)

```
ax.set_xlabel('Time (ps)')
plt.grid()
```



8.9 Production

[27]:
```
md_sys.production(out_folder=prod_folder,
                  nsteps=prod_step,
                  dt=dt, vsite=vsite, maxwarn=1)
```

```
gmx grompp -f prod_1Y0M.mdp -c ../sys_equi/sys_equi/02_equi_CA_LOW/equi_CA_LOW_1Y0M.gro -
↪r ../sys_equi/sys_equi/02_equi_CA_LOW/equi_CA_LOW_1Y0M.gro -p ../sys_top/1Y0M_water_
↪ion.top -po out_prod_1Y0M.mdp -o prod_1Y0M.tpr -maxwarn 1 -n ../sys_equi/sys_equi/02_
↪equi_CA_LOW/equi_CA_LOW_1Y0M.ndx
gmx mdrun -s prod_1Y0M.tpr -deffnm prod_1Y0M -nt 0 -ntmpi 0 -nsteps -2 -nocopyright
```

```
  0%|          | 0/5000000 [00:00<?, ?it/s]
```

8.10 Prepare trajectory

[40]:
```
# Center trajectory
md_sys.center_mol_box(traj=True)
```

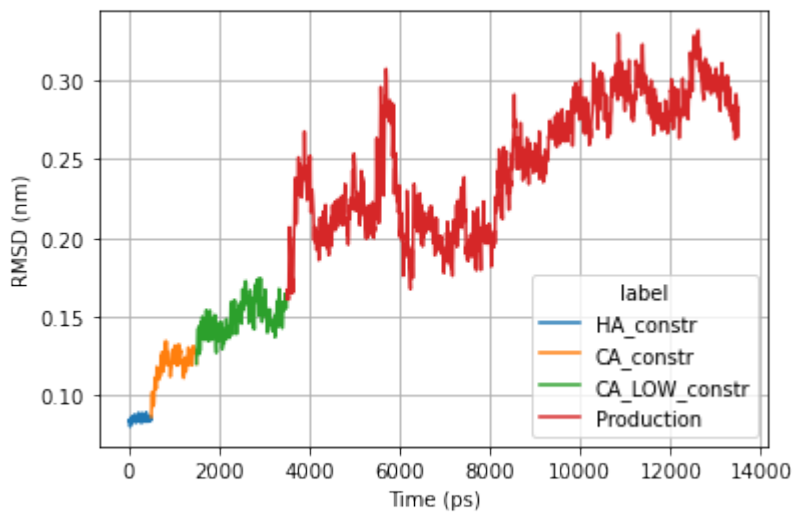
```
gmx make_ndx -f data_sim/sys_prod/prod_1Y0M.gro -o data_sim/sys_prod/prod_1Y0M.ndx
gmx trjconv -f data_sim/sys_prod/prod_1Y0M.xtc -o data_sim/sys_prod/prod_1Y0M_compact.
↪xtc -s data_sim/sys_prod/prod_1Y0M.tpr -ur tric -pbc mol -center yes -n data_sim/sys_
↪prod/prod_1Y0M.ndx
```

8.11 Basic Analysis

[44]:
```
rmsd_prod_pd = md_sys.get_rmsd(['C-alpha', 'Protein'], ref_sys=ref_sys)
rmsd_prod_pd['label'] = 'Production'

rmsd_prod_pd['time'] = rmsd_prod_pd['time'] + rmsd_pd['time'].max()
rmsd_all_pd = pd.concat([rmsd_pd, rmsd_prod_pd])
```

```
gmx rms -s data_sim/em/Init_em_1Y0M.tpr -f data_sim/sys_prod/prod_1Y0M_compact.xtc -n␣
↪data_sim/sys_prod/prod_1Y0M.ndx -o tmp_rmsd.xvg -fit rot+trans -ng 1 -pbc no
```

[48]:
```
ax = sns.lineplot(x="time", y="Protein",
                  hue="label",
                  data=rmsd_all_pd)
ax.set_ylabel('RMSD (nm)')
ax.set_xlabel('Time (ps)')
plt.grid()
```



8.12 Trajectory vizualisation

[58]:
```
# Align the protein coordinates
md_sys.convert_trj(select='Protein\nSystem\n', fit='rot+trans', pbc='none', skip='10')
```

```
gmx trjconv -f data_sim/sys_prod/prod_1Y0M_compact_compact.xtc -o data_sim/sys_prod/prod_
↪1Y0M_compact_compact_compact.xtc -s data_sim/sys_prod/prod_1Y0M.tpr -ur compact -pbc␣
↪none -fit rot+trans -n data_sim/sys_prod/prod_1Y0M.ndx -skip 10
```

[70]:
```
view = md_sys.view_traj()
view.add_representation(repr_type='licorice', selection='protein')
view.center(selection='CA')
view
```

```
NGLWidget(max_frame=100)
```

[4]:
```
# Unecessary, only need to nglview online:
IFrame(src='../_static/1Y0M_prod_traj.html', width=800, height=300)
```

[4]:
```
<IPython.lib.display.IFrame at 0x7fc6ccc5d730>
```

[ ]:
```

```

CHAPTER

NINE

SCRIPT

9.1 Tutorial

Here is an example of a short simulation of the SH3 domain of phospholipase C1. Seven successive steps are used:

1. Topologie creation using `create_top.py`.
2. Minimisation of the structure using `minimize_pdb.py`.
3. Solvation of the system using `solvate_ions.py`.
4. Minimisation of the system using `minimize_pdb.py`.
5. Equilibration of the system using `equi_3_step.py`.
6. Production run using `production.py`.
7. Extension of the production run using `extend.py`.

```
# Create topologie
create_top.py -f gromacs_py/test_files/1y0m.pdb -o tmp/1y0m/top -vsite

# Minimize the protein structure
minimize_pdb.py -f tmp/1y0m/top/1y0m_pdb2gmx_box.pdb -p tmp/1y0m/top/1y0m_pdb2gmx.top -o␣
↪tmp/1y0m/em/  -n em_1y0m -nt 2

# Add water and ions
solvate_ions.py -f tmp/1y0m/em/em_1y0m_compact.pdb -p tmp/1y0m/top/1y0m_pdb2gmx.top -o␣
↪tmp/1y0m_water_ions/top/  -n 1y0m_water_ions

# Minimize the system
minimize_pdb.py -f tmp/1y0m_water_ions/top/1y0m_water_ions_water_ion.gro -p tmp/1y0m_
↪water_ions/top/1y0m_water_ions_water_ion.top -o tmp/1y0m_water_ions/em/  -n em_1y0m

# Do three small equilibrations with postion contraints on heavy atoms (first), Carbon␣
↪alpha (second) and low constraint on Carbon alpha (third)
equi_3_step.py -f tmp/1y0m_water_ions/em/em_1y0m_compact.pdb -p tmp/1y0m_water_ions/top/
↪1y0m_water_ions_water_ion.top -o tmp/1y0m_water_ions/  -n 1y0m -HA_time 0.1 -dt_HA 0.
↪002 -CA_time 0.1 -CA_LOW_time 0.1 -dt 0.004 -maxwarn 1

# Small production run of 0.1 ns
production.py -f tmp/1y0m_water_ions/02_equi_CA_LOW/equi_CA_LOW_1y0m.gro -p tmp/1y0m_
↪water_ions/top/1y0m_water_ions_water_ion.top -o tmp/1y0m_water_ions/03_prod -n 1y0m -
↪time 0.1 -dt 0.004 -maxwarn 1
```

(continues on next page)

(continued from previous page)

```
# Extension of the simulation
extend.py -s tmp/1y0m_water_ions/03_prod/prod_1y0m.tpr -time 0.2

# Remove simulation files
rm -r ./tmp
```

Or simply use one command to do all previous commands:

```
top_em_equi_3_step_prod.py -f gromacs_py/test/input/1y0m.pdb -o tmp/1y0m -vsite -HA_time␣
↪0.1 -CA_time 0.1 -CA_LOW_time 0.1 -prod_time 0.3
```

9.2 Topologie related:

9.2.1 Create topologie

Create the topologie file from a structure pdb file

```
usage: create_top.py [-h] -f F -o O [-vsite]
```

Named Arguments

-f	Input PDB file
-o	Output directory
-vsite	Use virtual site for hydrogens

9.2.2 Solvate a system

Solvate a gromacs system with water and add ions to neutralize the system charge and to reach an ionic concentration

```
usage: solvate_ions.py [-h] -f F -p P -o O -n NAME [-d DIST] [-C CONC]
```

Named Arguments

-f	Input PDB file
-p	Topologie in gromacs format .top
-o	Output Directory
-n	Output file name
-d	Distance between the solute and the box
-C	Ion concentration (mM), default = 0.15 (150mM)

9.3 Simulation:

9.3.1 Energy minimization

Minimize a pdb structure in 2 steps,the first step without bonds constraints and the second step with

```
usage: minimize_pdb.py [-h] -f F -p P -o O -n NAME [-m_steps MIN_STEPS]
                       [-box BOX] [-nt NT] [-ntmpi NTMPI] [-gpu_id GPUID]
```

Named Arguments

-f	Input PDB file
-p	Topologie in gromacs format .top
-o	Output Directory
-n	Output file name
-m_steps	Minimisation nsteps, default=1000
-box	Create a box, default=False
-nt	Total number of threads to start, default=0
-ntmpi	Number of thread-MPI threads to start, default=0
-gpu_id	List of GPU device id-s to use, default=””

9.3.2 Equilibration

Equilibrate in 3 steps a system (coor+top), (i) first equilibration with heavy atoms position restraints, (ii) second equilibration with alpha carbon position restraints and (iii) finaly equilibration with weak alpha carbon position restraints

```
usage: equi_3_step.py [-h] -f F -p P -o O -n NAME [-HA_time HA_TIME]
                      [-CA_time CA_TIME] [-CA_LOW_time CA_LOW_TIME]
                      [-dt_HA DT_HA] [-dt DT] [-maxwarn MAXWARN] [-nt NT]
                      [-ntmpi NTMPI] [-gpu_id GPUID]
```

Named Arguments

-f	Input PDB file
-p	Topologie in gromacs format .top
-o	Output Directory
-n	Output file name
-HA_time	Equilibration with HA constraint time(ns), default = 0.25ns
-CA_time	Equilibration with HA constraint time(ns), default = 1ns
-CA_LOW_time	Equilibration with HA constraint time(ns), default = 5ns

-dt_HA	Equi HA dt, default=0.002 (2 fs)
-dt	Equi CA, CA_LOW, dt, default=0.002 (2 fs)
-maxwarn	Total number of warnings allowed for the equilibration, default=0
-nt	Total number of threads to start, default=0
-ntmpi	Number of thread-MPI threads to start, default=0
-gpu_id	List of GPU device id-s to use, default=””

9.3.3 Production

Simulation production

```
usage: production.py [-h] -f F -p P -o O -n NAME [-time TIME] [-dt DT]
                     [-maxwarn MAXWARN] [-nt NT] [-ntmpi NTMPI]
                     [-gpu_id GPUID]
```

Named Arguments

-f	Input PDB file
-p	Topologie in gromacs format .top
-o	Output Directory
-n	Output file name
-time	Production time, default=10
-dt	Equilibration dt, default=0.002 (2 fs)
-maxwarn	Total number of warnings allowed for the equilibration, default=0
-nt	Total number of threads to start, default=0
-ntmpi	Number of thread-MPI threads to start, default=0
-gpu_id	List of GPU device id-s to use, default=””

9.4 Peptide related:

9.4.1 Create peptide

Create a linear peptide structure, do a minimisation and a vacuum equilibration

```
usage: create_peptide.py [-h] -seq SEQ -o O [-m_steps MIN_STEPS] [-time TIME]
```

Named Arguments

-seq	Peptide sequence
-o	Output Directory
-m_steps	Minimisation nsteps, default=1000
-time	Vacuum equilibration time(ns), default = 1ns

9.4.2 Minimize cyclic peptide

Minimize a cyclic peptide structure in 2 steps, the first step without bonds constraints and the second step with bonds constraints

```
usage: minimize_pdb_and_cyclic.py [-h] -f F -n NAME [-dir OUT_DIR]
                                  [-m_steps MIN_STEPS] [-keep] [-cyclic]
                                  [-nt NT] [-ntmpi NTMPI] [-gpu_id GPUID]
                                  [-keep_segid] [-add_ter]
```

Named Arguments

-f	Input PDB file
-n	Output file name
-dir	Output directory for intermediate files
-m_steps	Minimisation nsteps, default=1000
-keep	Flag to keep temporary files (without flag output directory will be delete
-cyclic	Flag to indicate if the peptide/protein is cyclic
-nt	Total number of threads to start, default=0
-ntmpi	Number of thread-MPI threads to start, default=0
-gpu_id	List of GPU device id-s to use, default=””
-keep_segid	Flag to indicate if the original chain/segid should be kept
-add_ter	Flag to indicate if TER line should be included between chains and if residues in the input pdb file have non-consecutive residues

9.4.3 Insert n copy of a peptide in a system

Create a peptide strucure, insert it around a protein and do the minimisation, equilibration and production.

```
usage: sim_pep_prot.py [-h] -seq SEQ -npep NUM_MOL [-Pep_time PEP_TIME] -fsys
                       F_SYS -psys P_SYS -ssys S_SYS -o O -n NAME [-dt DT]
                       [-em_steps EM_STEPS] [-dt_HA DT_HA] [-HA_time HA_TIME]
                       [-CA_time CA_TIME] [-CA_LOW_time CA_LOW_TIME]
                       [-maxwarn MAXWARN] [-PROD_time PROD_TIME] [-nt NT]
                       [-ntmpi NTMPI] [-gpu_id GPUID]
```

Named Arguments

-seq	Peptide sequence
-npep	Number of molecule to insert
-Pep_time	Peptide vacuum equilibration with no constraint time(ns), default = 1ns
-fsys	Input PDB file of the system
-psys	Topologie in gromacs format .top of the system
-ssys	Input tpr file of the system
-o	Output Directory
-n	Output file name
-dt	Equilibration dt, default=0.005 (5 fs)
-em_steps	Minimisation steps, default = 5000
-dt_HA	Equilibration dt, default=0.005 (5 fs)
-HA_time	Equilibration with HA constraint time(ns), default = 0.25ns
-CA_time	Equilibration with HA constraint time(ns), default = 1ns
-CA_LOW_time	Equilibration with HA constraint time(ns), default = 5ns
-maxwarn	Total number of warnings allowed for the equilibration, default=0
-PROD_time	Production time(ns), default = 100ns
-nt	Total number of threads to start, default=0
-ntmpi	Number of thread-MPI threads to start, default=0
-gpu_id	List of GPU device id-s to use, default=””

~/Documents/repository/gromacs_py/equi_3_step.py -f 1y0m_water_ions/em/em_1y0m_compact.pdb -p 1y0m_water_ions/top/1y0m_water_ions_water_ion.top -o 1y0m_water_ions/ equi/ -n 1y0m -HA_time 0.1 -CA_time 0.2 -CA_LOW_time 0.4

CHAPTER

TEN

CREDITS

10.1 Development Lead

- Samuel Murail, Université de Paris <samuel.murail@u-paris.fr>

10.2 Contributors

- Maxence Delaunay
- Damien Espana

We are open to any contribution.

CHAPTER

ELEVEN

HISTORY

11.1 1.0.0 (2019-05-20)

- First release on PyPI.

11.2 1.2.0 (2019-05-20)

- Add simulation progress bar as a default feature.
- Add cystein disulfide bond modification of topologie.

CHAPTER

TWELVE

CONTRIBUTING

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

12.1 Types of Contributions

12.1.1 Report Bugs

Report bugs at https://github.com/samuelmurail/gromacs_py/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

12.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" and "help wanted" is open to whoever wants to implement it.

12.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "enhancement" and "help wanted" is open to whoever wants to implement it.

12.1.4 Write Documentation

Docking Python could always use more documentation, whether as part of the official Docking Python docs, in docstrings, or even on the web in blog posts, articles, and such.

12.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/samuelmurail/gromacs_py/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

12.2 Get Started!

Ready to contribute? Here's how to set up *gromacs_py* for local development.

1. Fork the *gromacs_py* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/gromacs_py.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv gromacs_py
$ cd gromacs_py/
$ python setup.py develop
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 gromacs_py tests
$ pytest
$ tox
```

   To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

12.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 3.5, 3.6, 3.7 and 3.8, and for PyPy. Check https://travis-ci.com/samuelmurail/gromacs_py/pull_requests and make sure that the tests pass for all supported Python versions.

12.4 Tips

To run a subset of tests:

```
$ pytest tests.test_gromacs_py
```

12.5 Deploying

A reminder for the maintainers on how to deploy. Make sure all your changes are committed (including an entry in HISTORY.rst). Then run:

```
$ bump2version patch # possible: major / minor / patch
$ git push
$ git push --tags
```

Travis will then deploy to PyPI if tests pass.

CHAPTER

THIRTEEN

GROMACS_PY

13.1 gromacs_py package

13.1.1 Subpackages

gromacs_py.gmx package

Submodules

gromacs_py.gmx.gmxsys module

class gromacs_py.gmx.gmxsys.**GmxSys**(*name=None*, *coor_file=None*, *top_file=None*, *tpr=None*)

Bases: object

Gromacs system encapsulation class.

This class can be used to launch most of gromacs commands (pdb2gmx, grompp, mdrun, trjconv, editconf, genconf, ...). After each steps, outputs file paths of gromacs commands are store in the class variable, like corr_file, top_file, tpr ... Most function will need the corr_file and/or top_file variable to be defined.

The GmxSys object can be considered as a md simulation system. Each operation on the object will affect the object variables.

The variables nt, ntmpi and gpu_id are only used by functions which run simulations run_simulation() like em() or production().

Parameters

- **name** (*str*) – generic name of the system
- **sim_name** (*str, optional*) – name of the simulation (used to create .tpr .log .edr ...)
- **coor_file** (*str, optional*) – path of the coordinate file (.pdb, .gro)
- **top_file** (*str, optional*) – path of the .top file
- **tpr** (*str, optional*) – path of the .tpr file
- **mdp** (*str, optional*) – path of the .mdp file
- **xtc** (*str, optional*) – path of the .xtc file
- **edr** (*str, optional*) – path of the .edr file
- **log** (*str, optional*) – path of the .log file
- **ndx** (*str, optional*) – path of the .ndx file

- **nt** (*int, default=0*) – Total number of threads to start
- **ntmpi** (*int, default=0*) – Number of thread-MPI threads to start
- **gpu_id** (*str, default=None*) – List of GPU device id-s to use, specifies the per-node PP rank to GPU mapping
- **sys_history** (*list of GmxSys()*) – List of previous GmxSys() states

Example

```
> show_log()
> TEST_OUT = str(getfixture('tmpdir'))
> prot = GmxSys(name='1y0m', coor_file=TEST_PATH+'/1y0m.pdb')
> ##################################
> ####   Create the topologie:   ###
> ##################################
> prot.prepare_top(out_folder=os.path.join(TEST_OUT, 'top_SH3'),     vsite=
↪'hydrogens') #doctest: +ELLIPSIS
Succeed to read file .../test_files/1y0m.pdb ,  648 atoms found
Succeed to read file .../test_files/1y0m.pdb ,  648 atoms found
Succeed to save file tmp_pdb2pqr.pdb
pdb2pqr30... --ff CHARMM --ffout CHARMM --keep-chain --titration-state-
↪method=propka --with-ph=7.00 tmp_pdb2pqr.pdb 00_1y0m.pqr
Succeed to read file 00_1y0m.pqr ,  996 atoms found
Chain: A  Residue: 0 to 60
Succeed to save file 01_1y0m_good_his.pdb
- Create topologie
gmx pdb2gmx -f 01_1y0m_good_his.pdb -o 1y0m_pdb2gmx.pdb -p     1y0m_pdb2gmx.top -i␣
↪1y0m_posre.itp -water tip3p -ff charmm36-jul2017     -ignh -vsite hydrogens
Molecule topologie present in 1y0m_pdb2gmx.top , extract the topologie in a␣
↪separate file: 1y0m_pdb2gmx.itp
Protein_chain_A
- ITP file: 1y0m_pdb2gmx.itp
- molecules defined in the itp file:
* Protein_chain_A
Rewrite topologie: 1y0m_pdb2gmx.top
> ##################################
> ####    Add water and ions:     ###
> ##################################
> prot.solvate_add_ions(out_folder=os.path.join(TEST_OUT, 'top_sys'))
↪#doctest: +ELLIPSIS
- Create pbc box
gmx editconf -f .../top_SH3/1y0m_pdb2gmx.pdb -o     .../top_SH3/1y0m_pdb2gmx_box.
↪pdb -bt dodecahedron -d 1.1
- Solvate the pbc box
Copy topologie file and dependancies
Copy topologie file and dependancies
- Create the tpr file genion_1y0m_water_ion.tpr
gmx grompp -f .../template/mini.mdp -c 1y0m_water.pdb -r     1y0m_water.pdb -p 1y0m_
↪water_ion.top -po out_mini.mdp -o     genion_1y0m_water_ion.tpr -maxwarn 1
- Add ions to the system with an ionic concentration of 0.15 M ,     sytem charge =␣
↪0.0 water num= 4775
Add ions : NA : 13   CL : 13
gmx genion -s genion_1y0m_water_ion.tpr -p 1y0m_water_ion.top -o     1y0m_water_ion.
↪gro -np 13 -pname NA -nn 13 -nname CL
```

(continues on next page)

(continued from previous page)

```
> ##################################
> ####    Minimize the system     ###
> ##################################
> prot.em(out_folder=os.path.join(TEST_OUT, 'em_SH3'), nsteps=10,     constraints=
↪'none')
- Create the tpr file 1y0m.tpr
gmx grompp -f 1y0m.mdp -c ../top_sys/1y0m_water_ion.gro -r     ../top_sys/1y0m_
↪water_ion.gro -p ../top_sys/1y0m_water_ion.top -po     out_1y0m.mdp -o 1y0m.tpr -
↪maxwarn 1
- Launch the simulation 1y0m.tpr
gmx mdrun -s 1y0m.tpr -deffnm 1y0m -nt 0 -ntmpi 0 -nsteps -2 -nocopyright
> ##################################
> ####    Create a D peptide      ###
> ##################################
> pep = GmxSys(name='D')
> pep.create_peptide(sequence='D', out_folder=os.path.join(TEST_OUT,     'top_D'),␣
↪em_nsteps=10, equi_nsteps=0, vsite='hydrogens')
-Make peptide: D
residue name:X
residue name:D
Succeed to save file .../top_D/D.pdb
- Create topologie
gmx pdb2gmx -f ../D.pdb -o D_pdb2gmx.pdb -p D_pdb2gmx.top -i D_posre.itp -water␣
↪tip3p -ff charmm36-jul2017 -ignh -ter -vsite hydrogens
Molecule topologie present in D_pdb2gmx.top , extract the topologie     in a␣
↪separate file: D_pdb2gmx.itp
Protein_chain_P
- ITP file: D_pdb2gmx.itp
- molecules defined in the itp file:
* Protein_chain_P
Rewrite topologie: D_pdb2gmx.top
- Create pbc box
gmx editconf -f .../top_D/00_top/D_pdb2gmx.pdb -o     .../top_D/00_top/D_pdb2gmx_
↪box.pdb -bt dodecahedron -d 1.0
- Create the tpr file D.tpr
gmx grompp -f D.mdp -c ../00_top/D_pdb2gmx_box.pdb -r     ../00_top/D_pdb2gmx_box.
↪pdb -p ../00_top/D_pdb2gmx.top -po out_D.mdp -o     D.tpr -maxwarn 1
- Launch the simulation D.tpr
gmx mdrun -s D.tpr -deffnm D -nt 0 -ntmpi 0 -nsteps -2 -nocopyright
> ######################################################
> ### Insert 4 copy of the peptide in the SH3 system: ###
> ######################################################
> prot.insert_mol_sys(mol_gromacs=pep, mol_num=4, new_name='SH3_D',     out_
↪folder=os.path.join(TEST_OUT, 'top_D_SH3')) #doctest: +ELLIPSIS
- Convert trj/coor
gmx trjconv -f ...D.gro -o ...D_compact.pdb -s ...D.gro -ur compact     -pbc none
Succeed to read file ...D_compact.pdb ,  22 atoms found
- Copy pbc box using genconf
Succeed to read file ...D_compact_copy_box.pdb ,  88 atoms found
Succeed to save file ...D_compact_copy_box.pdb
Res num: 8
- Convert trj/coor
```

(continues on next page)

(continued from previous page)

```
gmx trjconv -f ../em_SH3/1y0m.gro -o ../em_SH3/1y0m_compact.pdb -s     ../em_SH3/
↪1y0m.tpr -ur compact -pbc mol
Concat files: ['../em_SH3/1y0m_compact.pdb',     '../top_D/01_mini/D_compact_copy_
↪box.pdb']
Succeed to save concat file:  SH3_D_pre_mix.pdb
Succeed to read file SH3_D_pre_mix.pdb ,  15425 atoms found
Insert mol in system
Residue list = [4836, 4837, 4838, 4839, 4840, 4841, 4842, 4843]
Insert 4 mol of 2 residues each
insert mol   1, water mol   ..., time=0...
Warning atom 1MCH mass could not be founded
Warning atom 2MCH mass could not be founded
Warning atom 1HH3 mass could not be founded
Warning atom 2HH3 mass could not be founded
Warning atom 3HH3 mass could not be founded
insert mol   2, water mol   ..., time=0...
Warning atom 1MCH mass could not be founded
Warning atom 2MCH mass could not be founded
Warning atom 1HH3 mass could not be founded
Warning atom 2HH3 mass could not be founded
Warning atom 3HH3 mass could not be founded
insert mol   3, water mol   ..., time=0...
Warning atom 1MCH mass could not be founded
Warning atom 2MCH mass could not be founded
Warning atom 1HH3 mass could not be founded
Warning atom 2HH3 mass could not be founded
Warning atom 3HH3 mass could not be founded
insert mol   4, water mol   ..., time=0...
Warning atom 1MCH mass could not be founded
Warning atom 2MCH mass could not be founded
Warning atom 1HH3 mass could not be founded
Warning atom 2HH3 mass could not be founded
Warning atom 3HH3 mass could not be founded
Delete ... overlapping water atoms
Succeed to save file SH3_D.pdb
Ligand
Add 4 mol ...D_pdb2gmx.itp
Succeed to read file SH3_D.pdb ,  15... atoms found
Water num: 47...
[{'name': 'Protein_chain_A', 'num': '1'}, {'name': 'SOL', ...    {'name': 'Ligand',
↪'num': '4'}]
CHARGE: -4.0
Should neutralize the system
Copy topologie file and dependancies
- Create the tpr file genion_SH3_D_neutral.tpr
gmx grompp -f .../template/mini.mdp -c SH3_D.pdb -r SH3_D.pdb -p     SH3_D_neutral.
↪top -po out_mini.mdp -o genion_SH3_D_neutral.tpr -maxwarn 1
- Add ions to the system with an ionic concentration of 0 M ,     sytem charge = -4.
↪0 water num= 47...
Add ions : NA : 4   CL : 0
gmx genion -s genion_SH3_D_neutral.tpr -p SH3_D_neutral.top -o     SH3_D_neutral.
↪gro -np 4 -pname NA -nn 0 -nname CL
```

(continues on next page)

(continued from previous page)

```
> ################################
> ####   Minimize the system   ###
> ################################
> prot.em_2_steps(out_folder=os.path.join(TEST_OUT, 'top_D_SH3'),     no_constr_
↪nsteps=10, constr_nsteps=10)
- Create the tpr file Init_em_1y0m.tpr
gmx grompp -f Init_em_1y0m.mdp -c SH3_D_neutral.gro -r SH3_D_neutral.gro -p SH3_D_
↪neutral.top -po out_Init_em_1y0m.mdp -o Init_em_1y0m.tpr -maxwarn 1
- Launch the simulation Init_em_1y0m.tpr
gmx mdrun -s Init_em_1y0m.tpr -deffnm Init_em_1y0m -nt 0 -ntmpi 0     -nsteps -2 -
↪nocopyright
- Create the tpr file 1y0m.tpr
gmx grompp -f 1y0m.mdp -c Init_em_1y0m.gro -r Init_em_1y0m.gro -p     SH3_D_neutral.
↪top -po out_1y0m.mdp -o 1y0m.tpr -maxwarn 1
- Launch the simulation 1y0m.tpr
gmx mdrun -s 1y0m.tpr -deffnm 1y0m -nt 0 -ntmpi 0 -nsteps -2 -nocopyright
> ##################################
> ####    Show system history    ###
> ##################################
> prot.display_history() #doctest: +ELLIPSIS
State -3:
<BLANKLINE>
name          : 1y0m
sim_name      : genion_1y0m_water_ion
coor_file     : .../top_sys/1y0m_water_ion.gro
top_file      : .../top_sys/1y0m_water_ion.top
tpr           : .../top_sys/genion_1y0m_water_ion.tpr
mdp           : ...template/mini.mdp
nt            : 0
ntmpi         : 0
sys_history   : 0
<BLANKLINE>
State -2:
<BLANKLINE>
name          : 1y0m
sim_name      : genion_SH3_D_neutral
coor_file     : .../top_D_SH3/SH3_D_neutral.gro
top_file      : .../top_D_SH3/SH3_D_neutral.top
tpr           : .../top_D_SH3/genion_SH3_D_neutral.tpr
mdp           : ...template/mini.mdp
xtc           : .../em_SH3/1y0m.trr
edr           : .../em_SH3/1y0m.edr
log           : .../em_SH3/1y0m.log
nt            : 0
ntmpi         : 0
sys_history   : 0
<BLANKLINE>
State -1:
<BLANKLINE>
name          : 1y0m
sim_name      : Init_em_1y0m
coor_file     : .../top_D_SH3/Init_em_1y0m.gro
```

(continues on next page)

(continued from previous page)

```
top_file     : .../top_D_SH3/SH3_D_neutral.top
tpr          : .../top_D_SH3/Init_em_1y0m.tpr
mdp          : .../top_D_SH3/Init_em_1y0m.mdp
xtc          : .../top_D_SH3/Init_em_1y0m.trr
edr          : .../top_D_SH3/Init_em_1y0m.edr
log          : .../top_D_SH3/Init_em_1y0m.log
nt           : 0
ntmpi        : 0
sys_history  : 0
<BLANKLINE>
> ##################################
> ####   Equilibrate the system   ###
> ##################################
> equi_template_mdp = os.path.join(GROMACS_MOD_DIRNAME,     "template/equi_vsites.
↪mdp")
> mdp_options = {'nsteps': 100, 'define': '-DPOSRES', 'dt': 0.001}
> prot.run_md_sim(out_folder=os.path.join(TEST_OUT, 'equi_HA_D_SH3'),     name=
↪"equi_HA_D_SH3", mdp_template=equi_template_mdp,                          mdp_
↪options=mdp_options)
- Create the tpr file equi_HA_D_SH3.tpr
gmx grompp -f equi_HA_D_SH3.mdp -c ../top_D_SH3/1y0m.gro -r     ../top_D_SH3/1y0m.
↪gro -p ../top_D_SH3/SH3_D_neutral.top -po     out_equi_HA_D_SH3.mdp -o equi_HA_D_
↪SH3.tpr -maxwarn 0
- Launch the simulation equi_HA_D_SH3.tpr
gmx mdrun -s equi_HA_D_SH3.tpr -deffnm equi_HA_D_SH3 -nt 0 -ntmpi 0     -nsteps -2 -
↪nocopyright
> prot.get_simulation_time() #doctest: +ELLIPSIS
- Get simulation time from : .../equi_HA_D_SH3/equi_HA_D_SH3.cpt
gmx check -f .../equi_HA_D_SH3/equi_HA_D_SH3.cpt
0.1
> prot.convert_trj(traj=False) #doctest: +ELLIPSIS
- Convert trj/coor
gmx trjconv -f .../equi_HA_D_SH3/equi_HA_D_SH3.gro -o     .../equi_HA_D_SH3/equi_HA_
↪D_SH3_compact.pdb -s     .../equi_HA_D_SH3/equi_HA_D_SH3.tpr -ur compact -pbc mol
> prot.display() #doctest: +ELLIPSIS
name         : 1y0m
sim_name     : equi_HA_D_SH3
coor_file    : .../equi_HA_D_SH3/equi_HA_D_SH3_compact.pdb
top_file     : .../top_D_SH3/SH3_D_neutral.top
tpr          : .../equi_HA_D_SH3/equi_HA_D_SH3.tpr
mdp          : .../equi_HA_D_SH3/equi_HA_D_SH3.mdp
xtc          : .../equi_HA_D_SH3/equi_HA_D_SH3.xtc
edr          : .../equi_HA_D_SH3/equi_HA_D_SH3.edr
log          : .../equi_HA_D_SH3/equi_HA_D_SH3.log
nt           : 0
ntmpi        : 0
sys_history  : 4
> ########################################
> ### Extract Potential Energy and Temp ###
> ########################################
> ener_pd = prot.get_ener(['Potential', 'Temp'])  #doctest: +ELLIPSIS
- Extract energy
```

(continues on next page)

(continued from previous page)

```
gmx energy -f .../equi_HA_D_SH3/equi_HA_D_SH3.edr -o tmp_edr.xvg
> ener_pd['Potential'].mean() #doctest: +ELLIPSIS
-2...
> rmsd_pd = prot.get_rmsd(['C-alpha', 'Protein'])  #doctest: +ELLIPSIS
- Extract RMSD
- Create the ndx file ...equi_HA_D_SH3.ndx
gmx make_ndx -f ...equi_HA_D_SH3_compact.pdb -o ...equi_HA_D_SH3.ndx
gmx rms -s ...equi_HA_D_SH3.tpr -f ...equi_HA_D_SH3.xtc -n ...    equi_HA_D_SH3.ndx␣
↪-o tmp_rmsd.xvg -fit rot+trans -ng 1 -pbc no
> rmsd_pd #doctest: +ELLIPSIS
   time ...Protein
0   0.0...
> rmsf_pd = prot.get_rmsf(['Protein'], res="yes")  #doctest: +ELLIPSIS
- Extract RMSF
gmx rmsf -s ...equi_HA_D_SH3.tpr -f ...equi_HA_D_SH3.xtc -n ...    equi_HA_D_SH3.
↪ndx -o tmp_rmsf.xvg -fit no -res yes
> rmsf_pd #doctest: +ELLIPSIS
    Residue    RMSF
0       791  ...
```

Note: An history of all command used could be saved.

add_disulfide_bonds(*res_list*, *out_folder*, *name=None*, *check_file_out=True*, *ff='charmm36-jul2017'*)

Add disulfide bonds to a single protein topologie. Topologie has to be computed before using this function. Set specifically which cystein residues need to be bonded: Example of res_list = [[4, 7], [10, 20]], will connect cystein residues 4 to 7 and 10 to 20.

Parameters

- **res_list** – list of list of cystein residues to be bonded
- **out_folder** (*str*) – path of the output file folder
- **name** (*str, optional, default=None*) – generic name of the system
- **check_file_out** (*bool, optional, default=True*) – flag to check or not if file has already been created. If the file is present then the command break.
- **ff** (*str, optional, default="charmm36-jul2017"*) – forcefield.

Object requirement(s):

- self.coor_file
- self.top_file

Object field(s) changed:

- self.coor_file
- self.top_file

Add the following terms:

- 1 Bond
  - SG-SG
- 2 Angle

  - SG-SG-CB
  - CB-SG-SG
- 7 Dihed
  - CA-CB-SG-SG
  - HB1-CB-SG-SG
  - HB2-CB-SG-SG
  - CB-SG-SG-CB
  - SG-SG-CB-HB1
  - SG-SG-CB-HB2
  - SG-SG-CB-CA

Example

```
>>> show_log()
>>> TEST_OUT = getfixture('tmpdir')
>>>
>>> # Measure s-s bond length:
>>> ss_coor = pdb_manip.Coor(TEST_PATH+'/1dn3_cys.pdb')
Succeed to read file .../1dn3_cys.pdb ,  144 atoms found
>>> cystein_s_index = ss_coor.get_index_selection({'name': ['SG'], 'res_name' :␣
↪['CYS']})
>>> print(cystein_s_index)
[85, 118]
>>> distance = pdb_manip.Coor.atom_dist(ss_coor.atom_dict[cystein_s_index[0]],␣
↪ss_coor.atom_dict[cystein_s_index[1]])
>>> print('S-S distance = {:.2f} Å'.format(distance))
S-S distance = 6.38 Å
>>> no_ss_pep = GmxSys(name='1dn3_cys', coor_file=TEST_PATH+'/1dn3_cys.pdb')
>>>
>>> #Basic usage :
>>> no_ss_pep.prepare_top(out_folder=os.path.join(str(TEST_OUT),'1dn3/top'))
Succeed to read file .../1dn3_cys.pdb ,  144 atoms found
Succeed to read file .../1dn3_cys.pdb ,  144 atoms found
Succeed to save file tmp_pdb2pqr.pdb
pdb2pqr30... --ff CHARMM --ffout CHARMM --keep-chain --titration-state-
↪method=propka --with-ph=7.00 tmp_pdb2pqr.pdb 00_1dn3_cys.pqr
Succeed to read file 00_1dn3_cys.pqr ,  231 atoms found
Chain: A  Residue: 0 to 14
Succeed to save file 01_1dn3_cys_good_his.pdb
- Create topologie
gmx pdb2gmx -f 01_1dn3_cys_good_his.pdb -o 1dn3_cys_pdb2gmx.pdb -p 1dn3_cys_
↪pdb2gmx.top -i 1dn3_cys_posre.itp -water tip3p -ff charmm36-jul2017 -ignh -
↪vsite none
Molecule topologie present in 1dn3_cys_pdb2gmx.top , extract the topologie in a␣
↪separate file: 1dn3_cys_pdb2gmx.itp
Protein_chain_A
- ITP file: 1dn3_cys_pdb2gmx.itp
- molecules defined in the itp file:
```

(continues on next page)

(continued from previous page)

```
* Protein_chain_A
Rewrite topologie: 1dn3_cys_pdb2gmx.top
>>> no_ss_pep.add_disulfide_bonds(res_list=[[9, 12]], out_folder=os.path.
↪join(str(TEST_OUT),'1dn3/top_ss'))
Succeed to read file .../1dn3/top/1dn3_cys_pdb2gmx.pdb ,  231 atoms found
Succeed to save file .../1dn3/top_ss/1dn3_cys_ss_bond.pdb
Read rtp file : .../charmm36-jul2017.ff/merged.rtp
Correct residue CYS2 atom SG   atom type S    to SM ...
Correct residue CYS2 atom SG   atom type S    to SM ...
Protein_chain_A
>>> no_ss_pep.em(out_folder=TEST_OUT+'/1dn3/em_ss/', nsteps=10, create_box_
↪flag=True)
- Create pbc box
gmx editconf -f ...1dn3/top_ss/1dn3_cys_ss_bond.pdb -o .../1dn3/top_ss/1dn3_cys_
↪ss_bond_box.pdb -bt dodecahedron -d 1.0
- Create the tpr file 1dn3_cys.tpr
gmx grompp -f 1dn3_cys.mdp -c ../top_ss/1dn3_cys_ss_bond_box.pdb -r ../top_ss/
↪1dn3_cys_ss_bond_box.pdb -p ../top_ss/1dn3_cys_ss_bond.top -po out_1dn3_cys.
↪mdp -o 1dn3_cys.tpr -maxwarn 1
- Launch the simulation 1dn3_cys.tpr
gmx mdrun -s 1dn3_cys.tpr -deffnm 1dn3_cys -nt 0 -ntmpi 0 -nsteps -2 -
↪nocopyright
>>> # Need to convert the gro to pdb:
>>> no_ss_pep.convert_trj(traj=False)
- Convert trj/coor
gmx trjconv -f .../em_ss/1dn3_cys.gro -o .../em_ss/1dn3_cys_compact.pdb -s .../
↪em_ss/1dn3_cys.tpr -ur compact -pbc mol
>>> # Measure s-s bond length:
>>> ss_coor = pdb_manip.Coor(no_ss_pep.coor_file)
Succeed to read file .../em_ss/1dn3_cys_compact.pdb ,  229 atoms found
>>> cystein_s_index = ss_coor.get_index_selection({'name': ['SG'], 'res_name' :␣
↪['CYS']})
>>> print(cystein_s_index)
[135, 189]
>>> distance = pdb_manip.Coor.atom_dist(ss_coor.atom_dict[cystein_s_index[0]],␣
↪ss_coor.atom_dict[cystein_s_index[1]])
>>> print('S-S distance = {:.2f} Å'.format(distance))
S-S distance = 2.0... Å
```

Note: No options are allowed (water model, termini capping) except for vsites.

add_ions(*out_folder*, *name=None*, *ion_C=0.15*, *pname='NA'*, *nname='CL'*, *solv_name='SOL'*, *maxwarn=1*, *check_file_out=True*)

Add ion in a system to neutralise the sys_charge and to reach the ionic concentration `ion_C`.

Ion number are computed using the water number and the charge of the system:

1. With $cation_{num} = \frac{int(C_{ion} * water_{num})}{55.5}$
2. if $cation_{num} + sys_{charge} >= 0$ then $anion_{num} = cation_{num} + sys_{charge}$ else $cation_{num} = -sys_{charge}$

Parameters

- **out_folder** (*str*) – path of the output file folder
- **name** (*str, optional, default=None*) – generic name of the system
- **ion_C** (*float, optional, default=0.15*) – ionic concentraton (Molar)
- **pname** (*str, optional, default="SOL"*) – cation name
- **pname** – anion name
- **pname** – solvant name
- **check_file_out** (*bool, optional, default=True*) – flag to check or not if file has already been created. If the file is present then the command break.

Object requirement(s):

- self.coor_file
- self.top_file

Object field(s) changed:

- self.coor_file
- self.top_file

Example

```
>>> TEST_OUT = getfixture('tmpdir')
>>> prot = GmxSys(name='1y0m', coor_file=TEST_PATH+'/1y0m.pdb')
>>> prot.add_top(out_folder=TEST_OUT+'/add_ions/top_SH3/')
- Create topologie
gmx pdb2gmx -f .../test_files/1y0m.pdb -o 1y0m_pdb2gmx.pdb -p 1y0m_pdb2gmx.top -
↪i 1y0m_posre.itp -water tip3p -ff charmm36-jul2017
Molecule topologie present in 1y0m_pdb2gmx.top , extract the topologie in a␣
↪separate file: 1y0m_pdb2gmx.itp
Protein_chain_A
- ITP file: 1y0m_pdb2gmx.itp
- molecules defined in the itp file:
* Protein_chain_A
Rewrite topologie: 1y0m_pdb2gmx.top
>>> prot.create_box()
- Create pbc box
gmx editconf -f .../add_ions/top_SH3/1y0m_pdb2gmx.pdb -o .../add_ions/top_SH3/
↪1y0m_pdb2gmx_box.pdb -bt dodecahedron -d 1.0
>>> prot.solvate_box(out_folder=TEST_OUT+'/add_ions/top_SH3_water/')
- Solvate the pbc box
Copy topologie file and dependancies
>>> prot.add_ions(out_folder=TEST_OUT+'/add_ions/top_SH3_water_ions/')
Copy topologie file and dependancies
- Create the tpr file genion_1y0m_ion.tpr
gmx grompp -f .../template/mini.mdp -c ../top_SH3_water/1y0m_water.pdb -r ../
↪top_SH3_water/1y0m_water.pdb -p 1y0m_ion.top -po out_mini.mdp -o genion_1y0m_
↪ion.tpr -maxwarn 1
- Add ions to the system with an ionic concentration of 0.15 M , sytem charge =␣
↪0.0 water num= 56...
```

(continues on next page)

(continued from previous page)

```
Add ions : NA : 15   CL : 15
gmx genion -s genion_1y0m_ion.tpr -p 1y0m_ion.top -o 1y0m_ion.gro -np 15 -pname␣
↪NA -nn 15 -nname CL
>>> prot.em(out_folder=TEST_OUT+'/add_ions/em_SH3_water_ions/', nsteps=10,␣
↪constraints="none")
- Create the tpr file 1y0m.tpr
gmx grompp -f 1y0m.mdp -c ../top_SH3_water_ions/1y0m_ion.gro -r ../top_SH3_
↪water_ions/1y0m_ion.gro -p ../top_SH3_water_ions/1y0m_ion.top -po out_1y0m.
↪mdp -o 1y0m.tpr -maxwarn 1
- Launch the simulation 1y0m.tpr
gmx mdrun -s 1y0m.tpr -deffnm 1y0m -nt 0 -ntmpi 0 -nsteps -2 -nocopyright
```

Note: If `name` is not defined, the command will create a new .pdb and .top file name after the object name and adding "_ion".

Note: There might be some charge issues with amber forcefield for example. There is a discrepency between the atom charge and the total charge column with amber. In the itp charge is chown with a 2 decimal precision as in the rtp file it can be up to 5 decimals. Should consider using the total charge to deduce the atom charge and avoid errors. Up to now it has been fixed using *round* function instead of *int* for system charge

add_mdp(*mdp_template*, *mdp_options*, *folder_out=''*, *check_file_out=True*)

Create the MD simulation input mdp file from a template.

Read a template mdp file and replace define fields in mdp_options with the new value. In case the field name has a '-' , repalce it by : '_'.

Parameters

- **mdp_template** (*str*) – mdp file template
- **mdp_options** (*dict*) – New parameters to use
- **folder_out** (*str, default=''*) – Path for output file
- **check_file_out** (*bool, optional, default=True*) – flag to check or not if file has already been created. If the file is present then the command break.

Object requirement(s):

- self.sim_name

Object field(s) changed:

- self.mdp

add_ndx(*ndx_cmd_input*, *ndx_name=None*, *folder_out=''*, *check_file_out=True*)

Create a ndx file using `gmx make_ndx`

Parameters

- **ndx_cmd_input** (*str*) – Input arguments for `gmx make_ndx`
- **ndx_name** (*str, default=None*) – output name for the index file
- **folder_out** (*str, default=''*) – Path for output file

- **check_file_out** (*bool, optional, default=True*) – flag to check or not if file has already been created. If the file is present then the command break.

Object requirement(s):

- self.coor_file

Object field(s) changed:

- self.ndx

Note: If name is not defined, will use the name of the object.

---

add_top(*out_folder, name=None, ff='charmm36-jul2017', water='tip3p', check_file_out=True, pdb2gmx_option_dict={}, input_pdb2gmx='', posre_post=''*)

Launch the pdb2gmx command.

The objet variable self.coor_file has to be defined before launching this function. `pdb2gmx` will create a new coordinate file `name+"_pdb2gmx.pdb"`, a topologie `name+"_pdb2gmx.top"` and several molecule itp and posre files. If name is not defined, it will use the object name.

Parameters

- **out_folder** (*str*) – path of the output file folder
- **name** (*str, optional, default=None*) – generic name of the system
- **ff** (*str, optional, default="charmm36"*) – forcefield
- **water** (*str, optional, default="tip3p"*) – water model
- **check_file_out** (*bool, optional, default=True*) – flag to check or not if file has already been created. If the file is present then the command break.
- **pdb2gmx_option_dict** (*dict, optional, default=None*) – dictionnary of option for pdb2gmx, for example if you want to ignore input hydrogens use: `{'ignh': None}`. The '-' before the option is to avoid.
- **input_pdb2gmx** (*str, optional, default=None*) – input for pdb2gmx request

Object requirement(s):

- self.coor_file

Object field(s) changed:

- self.coor_file
- self.top_file

Example

```
>>> TEST_OUT = getfixture('tmpdir')
>>> prot = GmxSys(name='1y0m', coor_file=TEST_PATH+'/1y0m.pdb')
>>> #Basic usage :
>>> prot.add_top(out_folder=TEST_OUT+'/add_top/top_SH3')
- Create topologie
gmx pdb2gmx -f .../test_files/1y0m.pdb -o 1y0m_pdb2gmx.pdb -p 1y0m_pdb2gmx.top -
↪i 1y0m_posre.itp -water tip3p -ff charmm36-jul2017
Molecule topologie present in 1y0m_pdb2gmx.top , extract the topologie in a␣
↪separate file: 1y0m_pdb2gmx.itp
```

(continues on next page)

(continued from previous page)

```
Protein_chain_A
- ITP file: 1y0m_pdb2gmx.itp
- molecules defined in the itp file:
* Protein_chain_A
Rewrite topologie: 1y0m_pdb2gmx.top
>>> ##########################################
>>> # Use of different options for pdb2gmx: #
>>> ##########################################
>>> # Ignore hydrogens: 'ignh': None
>>> # Define amino acid termini: 'ter': None
>>> # Needs to answer pdb2gmx request concerning termini
>>> # with: input_pdb2gmx ="1 \n 0"
>>> prot = GmxSys(name='1y0m', coor_file=TEST_PATH+'/1y0m.pdb')
>>> prot.add_top(out_folder=TEST_OUT+'/add_top/top_SH3_2/',
...     pdb2gmx_option_dict={'ignh': None, 'ter': None},
...     input_pdb2gmx="1 \n 0")
- Create topologie
gmx pdb2gmx -f .../test_files/1y0m.pdb -o 1y0m_pdb2gmx.pdb -p 1y0m_pdb2gmx.top -
↪i 1y0m_posre.itp -water tip3p -ff charmm36-jul2017 -ignh -ter
Molecule topologie present in 1y0m_pdb2gmx.top , extract the topologie in a␣
↪separate file: 1y0m_pdb2gmx.itp
Protein_chain_A
- ITP file: 1y0m_pdb2gmx.itp
- molecules defined in the itp file:
* Protein_chain_A
Rewrite topologie: 1y0m_pdb2gmx.top
```

Note: To avoid conflict with focefields, the environment variable `$GMXLIB` is change to `GROMACS_MOD_DIRNAME+"/template/"` where curently only charmm36 is present, if you want to use another forcefield, copy your forcefield folder in `GROMACS_MOD_DIRNAME+"/template/"`, or change the current code.

add_tpr(*name*, *r=None*, *po=None*, *folder_out=''*, *check_file_out=True*, ***grompp_options*)

Create a tpr file using `gmx grompp`.

Parameters

- **name** (*str*) – name of the simulation
- **r** (*str, default=None*) – reference coordinate file for position restraints
- **po** (*str, default=None*) – output file for the mdp file
- **folder_out** (*str, default=""*) – Path for output file
- **check_file_out** (*bool, optional, default=True*) – flag to check or not if file has already been created. If the file is present then the command break.
- ****grompp_options** – Optional arguments for `gmx grompp`

Object requirement(s):

- self.mdp
- self.coor_file

- self.top_file

Object optional requirement(s):

- self.ndx

Object field(s) changed:

- self.tpr

center_mol_box(*sele_dict={'name': ['N', 'C', 'O', 'CA', 'CB', 'CG', 'CG1', 'CG2', 'SG', 'OG', 'OG1', 'CD', 'CD1', 'CD2', 'OD1', 'OD2', 'SD', 'ND1', 'CE', 'CE1', 'CE2', 'CE3', 'OE1', 'OE2', 'NE', 'NE1', 'NE2', 'OH', 'CZ', 'CZ2', 'CZ3', 'NZ', 'NH1', 'NH2', "O5'", "C5'", "C4'", "O4'", "C1'", 'N1', 'C6', 'CG2', 'C5', 'C4', 'N4', 'N3', 'C2', 'O2', "C3'", "C2'", "O3'", 'P', 'O1P', 'O2P', 'N9', 'C8', 'N7', 'O6', 'N2', 'C7', 'N6', 'O4'], 'res_name': ['GLY', 'HIS', 'HSP', 'HSE', 'HSD', 'HIP', 'HIE', 'HID', 'ARG', 'LYS', 'ASP', 'ASPP', 'ASN', 'GLU', 'GLUP', 'GLN', 'SER', 'THR', 'ASN', 'GLN', 'CYS', 'SEC', 'PRO', 'ALA', 'ILE', 'PHE', 'TYR', 'TRP', 'VAL', 'LEU', 'MET', 'DA5', 'DA3', 'DAN', 'DA', 'DT5', 'DT3', 'DTN', 'DT', 'DC5', 'DC3', 'DCN', 'DC', 'DG5', 'DG3', 'DGN', 'DG', 'RA5', 'RA3', 'RAN', 'RA', 'RU5', 'RU3', 'RUN', 'RU', 'RC5', 'RC3', 'RCN', 'RC', 'RG5', 'RG3', 'RGN', 'RG']}*, *traj=False*, *ref_coor=None*, ***cmd_args*)

Center a sytem on a selection of residue

Parameters
res_list (*str*) – List of residues

static concat_coor(**coor_in_files*, *pdb_out*, *check_file_out=True*)

Concat a list of coordinates file in one coordinate file:

Parameters

- **coor_in_files** (*list of str*) – list of pdb/gro files
- **pdb_out** (*str*) – file to save the concat structure

Returns
name of the new pdb file

Return type
str

Note: This function does not use or affect the GmxSys object.

---

concat_edr(**edr_files_list*, *concat_edr_out*, *check_file_out=True*)

Concat a list of energy file in one energy file:

Parameters

- **xtc_in_files** (*list of str*) – list of edr files
- **concat_edr_out** (*str*) – file to save the concat energy

Object field(s) changed:

- self.edr

concat_traj(**xtc_files_list*, *concat_traj_out*, *check_file_out=True*)

Concat a list of trajectory file in one trajectory file:

Parameters

- **xtc_in_files** (*list of str*) – list of xtc files

- **concat_traj_out** (*str*) – file to save the concat trajectory

Object field(s) changed:

- self.xtc

convert_selection_to_index(*selection_list*)

Convert selection list with selection name eg. ["System"] to the index number eg. ["0"].

Parameters
selection_list (*list*) – List of selection names

Returns
list of selection numbers

Return type
list

convert_trj(*name=None*, *ur='compact'*, *pbc='mol'*, *select='System'*, *traj=True*, *specific_coor_out=None*, *check_file_out=True*, *tpr=None*, ***cmd_args*)

Convert a trajectory or coordinate file using the commande `gmx trjconv`.

This is specially usefull when the protein is break across pbc. Using `convert_trj()` with default parameters will fix it.

Parameters

- **name** (*str, optional, default=None*) – generic name of the system
- **ur** (*str, default="compact"*) – unit-cell representation ("rect", "tric", "compact")
- **pbc** (*str, default="mol"*) – PBC treatment ("none", "mol", "res", "atom", "nojump", "cluster", "whole")
- **select** (*str, default="System"*) – group for output
- **specific_coor_out** (*str, optional, default=None*) – specific output file
- **traj** (*bool, default=True*) – Flag to convert trajectory or coordinates
- **check_file_out** (*bool, optional, default=True*) – flag to check or not if file has already been created. If the file is present then the command break.
- **cmd_args** – Optional arguments for `gmx trjconv`

Object requirement(s):

- self.tpr
- self.coor_file or self.xtc

Object field(s) changed:

- self.coor_file or self.xtc

Example

```
>>> TEST_OUT = getfixture('tmpdir')
>>> prot = GmxSys(name='1y0m', coor_file=TEST_PATH+'/1y0m.pdb')
>>> prot.add_top(out_folder=TEST_OUT+'/convert_trj/top_SH3/')
- Create topologie
gmx pdb2gmx -f .../test_files/1y0m.pdb -o 1y0m_pdb2gmx.pdb -p 1y0m_pdb2gmx.top -
↪i 1y0m_posre.itp -water tip3p -ff charmm36-jul2017
```

(continues on next page)

(continued from previous page)

```
Molecule topologie present in 1y0m_pdb2gmx.top , extract the topologie in a␣
↪separate file: 1y0m_pdb2gmx.itp
Protein_chain_A
- ITP file: 1y0m_pdb2gmx.itp
- molecules defined in the itp file:
* Protein_chain_A
Rewrite topologie: 1y0m_pdb2gmx.top
>>> prot.create_box()
- Create pbc box
gmx editconf -f .../convert_trj/top_SH3/1y0m_pdb2gmx.pdb -o .../convert_trj/top_
↪SH3/1y0m_pdb2gmx_box.pdb -bt dodecahedron -d 1.0
>>> prot.solvate_box(out_folder=TEST_OUT+'/convert_trj/top_SH3_water/')
- Solvate the pbc box
Copy topologie file and dependancies
>>> prot.em(out_folder=TEST_OUT+'/convert_trj/em_SH3_water/', nsteps=10,␣
↪constraints="none")
- Create the tpr file 1y0m.tpr
gmx grompp -f 1y0m.mdp -c ../top_SH3_water/1y0m_water.pdb -r ../top_SH3_water/
↪1y0m_water.pdb -p ../top_SH3_water/1y0m_water.top -po out_1y0m.mdp -o 1y0m.
↪tpr -maxwarn 1
- Launch the simulation 1y0m.tpr
gmx mdrun -s 1y0m.tpr -deffnm 1y0m -nt 0 -ntmpi 0 -nsteps -2 -nocopyright
>>> prot.convert_trj(traj=False)
- Convert trj/coor
gmx trjconv -f .../convert_trj/em_SH3_water/1y0m.gro -o .../convert_trj/em_SH3_
↪water/1y0m_compact.pdb -s .../convert_trj/em_SH3_water/1y0m.tpr -ur compact -
↪pbc mol
```

Note: If `name` is not defined, the command will create a new pdb file name after the input one and adding "_compact.pdb" or "_compact.xtc". If `name` is defined the pdb filed will be saved in the same directory as input file, the "_compact.pdb" or "_compact.xtc" will be added to `name`.

property coor_file

copy_box(*nbox*, *name=None*, *check_file_out=True*, *renumber=False*, ***cmd_args*)

Copy images of a given corrdinates in x, y, and z directions using `gmx genconf`.

nbox needs a list of 3 string for number x,y,z dimensions copy

This is specially usefull when the protein is break across pbc. Using `convert_trj()` with default parameters will fix it.

Parameters

- **nbox** (*list of string*) – list of 3 string for number of x, y, z dimensions copy
- **name** (*str, optional, default=None*) – generic name of the system
- **check_file_out** (*bool, optional, default=True*) – flag to check or not if file has already been created. If the file is present then the command break.
- **cmd_args** – Optional arguments for `gmx genconf`

Object requirement(s):

- self.coor_file

Object field(s) changed:

- self.coor_file

Example

```
>>> TEST_OUT = getfixture('tmpdir')
>>> prot = GmxSys(name='1y0m', coor_file=TEST_PATH+'/1y0m.pdb')
>>> prot.add_top(out_folder=TEST_OUT+'/copy_box/top_SH3/')
- Create topologie
gmx pdb2gmx -f .../test_files/1y0m.pdb -o 1y0m_pdb2gmx.pdb -p 1y0m_pdb2gmx.top -
↪i 1y0m_posre.itp -water tip3p -ff charmm36-jul2017
Molecule topologie present in 1y0m_pdb2gmx.top , extract the topologie in a␣
↪separate file: 1y0m_pdb2gmx.itp
Protein_chain_A
- ITP file: 1y0m_pdb2gmx.itp
- molecules defined in the itp file:
* Protein_chain_A
Rewrite topologie: 1y0m_pdb2gmx.top
>>> prot.create_box()
- Create pbc box
gmx editconf -f .../copy_box/top_SH3/1y0m_pdb2gmx.pdb -o .../copy_box/top_SH3/
↪1y0m_pdb2gmx_box.pdb -bt dodecahedron -d 1.0
>>> prot.copy_box(nbox=[4,1,1])
- Copy pbc box using genconf
```

Note: If `name` is not defined, the command will create a new pdb file name after the input one and adding "_copy_box.pdb". If `name` is defined the pdb filed will be saved in the same directory as input file, "_copy_box.pdb" will be added to `name`.

create_box(*name=None*, *dist=1.0*, *box_type='dodecahedron'*, *check_file_out=True*)

Create pbc box using `gmx editconf`

Parameters

- **name** (*str, optional, default=None*) – generic name of the system
- **dist** (*float, default=1.0*) – Distance between the solute and the box (nm)
- **box_type** (*str, default="dodecahedron"*) – Box type ("triclinic", "cubic", "dodecahedron", "octahedron")
- **check_file_out** (*bool, optional, default=True*) – flag to check or not if file has already been created. If the file is present then the command break.

Object requirement(s):

- self.coor_file

Object field(s) changed:

- self.coor_file

Example

```
>>> TEST_OUT = getfixture('tmpdir')
>>> prot = GmxSys(name='1y0m', coor_file=TEST_PATH+'/1y0m.pdb')
>>> prot.add_top(out_folder=TEST_OUT+'/create_box/top_SH3/')
- Create topologie
gmx pdb2gmx -f .../test_files/1y0m.pdb -o 1y0m_pdb2gmx.pdb -p 1y0m_pdb2gmx.top -
↪i 1y0m_posre.itp -water tip3p -ff charmm36-jul2017
Molecule topologie present in 1y0m_pdb2gmx.top , extract the topologie in a␣
↪separate file: 1y0m_pdb2gmx.itp
Protein_chain_A
- ITP file: 1y0m_pdb2gmx.itp
- molecules defined in the itp file:
* Protein_chain_A
Rewrite topologie: 1y0m_pdb2gmx.top
>>> prot.create_box()
- Create pbc box
gmx editconf -f .../create_box/top_SH3/1y0m_pdb2gmx.pdb -o .../create_box/top_
↪SH3/1y0m_pdb2gmx_box.pdb -bt dodecahedron -d 1.0
```

Note: If `name` is not defined, the command will create a new pdb file name after the input one and adding "_box.pdb". If `name` is defined the pdb filed will be saved in the same directory as input file, the "_box.pdb" will be added to `name`.

create_itp_atomtype_ion_octa_dummy(*atomtypes*, *ion_name=['MN', 'ZN']*)

Forcefield A and B values taken from : Duarte et al. 2014 J. Phys. Chem. B

https://en.wikipedia.org/wiki/Lennard-Jones_potential

$A = 4\epsilon\sigma^{12}$

$B = 4\epsilon\sigma^{6}$

$\sigma = \sqrt[6]{\frac{A}{B}}$

$\epsilon = \frac{B^2}{4A}$

ion_name=['NI', 'CO', 'ZN', 'MN', 'FE', 'MG', 'CA']

; MM 171 35 ; D 0.05 0 ; ; MN ; C12, C6 = 171**2, 35**2 ; sigma = (C12/C6)**(1/6) = 1.697 A = 0.1697 nm ; eps = C6**2/(4*C12) = 12.829 Kcal mol-1 = 418.4 KJ mol -1 ; ; DMN ; C12, C6 = 0, 35**2

; MN 25 36.938000 0.000 A 0.1697 12.829 ; Gromacs unit is : kJ mol1 nm2 kb = kb_cal_A * 4.184 * 100

; Aqvist and Warshel JACS 1990 ; ; MM 145 25 ; D 0 0 ; Kb = 1600 (kcal mol1Å2) and K = 250 (kcal mol1rad2) and ; no bond between dummies. ; C12, C6 = 145**2, 25**2 ; sigma = (C12/C6)**(1/6) = 1.7967 A = 0.17967 nm ; eps = C6**2/(4*C12) = 4.645 Kcal mol-1 = 19.4337 KJ mol -1 ; MN 25 36.938000 0.000 A 0.17967 19.4337

create_itp_ion_octa_dummy(*atomtypes*, *ion_name=['MN', 'ZN']*)

create_mdp(*mdp_options*, *folder_out=''*, *check_file_out=True*)

Create the MD simulation input mdp file.

Parameters

- **mdp_options** (*dict*) – New parameters to use
- **folder_out** (*str, default=""*) – Path for output file

- **check_file_out** (*bool, optional, default=True*) – flag to check or not if file has already been created. If the file is present then the command break.

Object requirement(s):

- self.sim_name

Object field(s) changed:

- self.mdp

create_peptide(*sequence, out_folder, N_ter='None', C_ter='COOH', em_nsteps=1000, equi_nsteps=10000, posre_post='_pep', vsite='none'*)

Create a linear peptide structure and topologie:

1. **Create a peptide with pymol with one more residue G at the** beginning of the peptide. This residue will then be change to an ACE. NH2 terminaison raise some issue with virtual sites and cannot be used.
2. Create the topologie using `add_top()`
3. Minimise the structure using `em()`
4. Do a vacuum equilibration of the peptide using `run_md_sim()`

Parameters

- **out_folder** (*str*) – path of the output file folder
- **name** (*str, optional, default=None*) – generic name of the system
- **ion_C** (*float, optional, default=0.15*) – ionic concentraton (Molar)
- **vsite** (*str, optional, default="none"*) – option for topologie's bonds constraints ("none", "hydrogens", "all")

Object requirement(s):

- None

Object field(s) changed:

- self.coor_file
- self.top_file

Example

```
> pep = GmxSys(name='SAM_pep')
> pep.create_peptide(sequence='SAM', out_folder=os.path.join(str(TEST_OUT),
↪'peptide'), em_nsteps=10, equi_nsteps=10, vsite='hydrogens')
-Make peptide: SAM
residue name:X
residue name:S
residue name:A
residue name:M
Succeed to save file .../peptide/SAM.pdb
- Create topologie
gmx pdb2gmx -f ../SAM.pdb -o SAM_pdb2gmx.pdb -p SAM_pdb2gmx.top -i SAM_posre.
↪itp -water tip3p -ff charmm36-jul2017 -ignh -ter -vsite hydrogens
Molecule topologie present in SAM_pdb2gmx.top , extract the topologie in a␣
↪separate file: SAM_pdb2gmx.itp
```

(continues on next page)

(continued from previous page)

```
Protein_chain_P
- ITP file: SAM_pdb2gmx.itp
- molecules defined in the itp file:
* Protein_chain_P
Rewrite topologie: SAM_pdb2gmx.top
- Create pbc box
gmx editconf -f .../peptide/00_top/SAM_pdb2gmx.pdb -o .../peptide/00_top/SAM_
↪pdb2gmx_box.pdb -bt dodecahedron -d 1.0
- Create the tpr file SAM_pep.tpr
gmx grompp -f SAM_pep.mdp -c ../00_top/SAM_pdb2gmx_box.pdb -r ../00_top/SAM_
↪pdb2gmx_box.pdb -p ../00_top/SAM_pdb2gmx.top -po out_SAM_pep.mdp -o SAM_pep.
↪tpr -maxwarn 1
- Launch the simulation SAM_pep.tpr
gmx mdrun -s SAM_pep.tpr -deffnm SAM_pep -nt 0 -ntmpi 0 -nsteps -2 -nocopyright
- Create the tpr file equi_vacuum_SAM.tpr
gmx grompp -f equi_vacuum_SAM.mdp -c ../01_mini/SAM_pep.gro -r ../01_mini/SAM_
↪pep.gro -p ../00_top/SAM_pdb2gmx.top -po out_equi_vacuum_SAM.mdp -o equi_
↪vacuum_SAM.tpr -maxwarn 1
- Launch the simulation equi_vacuum_SAM.tpr
gmx mdrun -s equi_vacuum_SAM.tpr -deffnm equi_vacuum_SAM -nt 0 -ntmpi 0 -nsteps␣
↪-2 -nocopyright
```

Warning: The peptide function won't work with gromacs version above 2018. There is issues with COOH C-temini, see: https://redmine.gromacs.org/issues/3301 Use another C-ter or use a previous version of gromacs.

cyclic_peptide_top(*out_folder*, *name=None*, *check_file_out=True*, *ff='charmm36-jul2017'*)

Prepare a topologie for a cyclic peptide

1. Create a peptide topologie with NH3+ Cter and COO- Nter using `add_top()`. 2. Delete useless termini atoms. 3. Change atom types, names and charges. 4. Add backbone bonds, angle and dihedral parameters. 5. Finally compute the topologie with pdb2gmx add_top()

Parameters

- **out_folder** (*str*) – path of the output file folder
- **name** (*str, optional, default=None*) – generic name of the system
- **check_file_out** (*bool, optional, default=True*) – flag to check or not if file has already been created. If the file is present then the command break.

Object requirement(s):

- self.coor_file

Object field(s) changed:

- self.coor_file
- self.top_file

Example

Note: No options are allowed (water model, termini capping) except for vsites.

Warning: Has not been tested with special residues like GLY or PRO !!

display()

Display defined attribute of the GmxSys object.

display_history()

Show all history

property edr

em(*out_folder*, *name=None*, *posres=''*, *create_box_flag=False*, *monitor_tool={'file_check_ext': 'log', 'function': <function progress_bar>}*, *maxwarn=1*, ***mdp_options*)

Minimize a system.

Parameters

- **out_folder** (*str*) – path of the output file folder
- **name** (*str, default=None*) – name of the simulation to run
- **nsteps** (*int, default=1000*) – number of minimisation steps
- **posres** (*str, default=''*) – option for the `define` variable in the mdp file, need to be define to have postion restraints.
- **create_box_flag** – flag to create or not a box to the input coor file.
- **maxwarn** (*int, default=0*) – Maximum number of warnings when using `gmx grompp`
- **monitor** (*dict, default=None*) – option to monitor a simulation, if not none monitor should contains two values: `function` the function to be ran while simulation is running and `input` parameters for the function
- **mdp_options** (*dict*) – Additional mdp parameters to use

Object requirement(s):

- self.coor_file
- self.top_file
- self.nt
- self.ntmpi
- self.gpu_id

Object field(s) changed:

- self.mdp
- self.tpr
- self.coor_file
- self.xtc

em_2_steps(*out_folder*, *name=None*, *no_constr_nsteps=1000*, *constr_nsteps=1000*, *posres=''*, *create_box_flag=False*, *monitor_tool={'file_check_ext': 'log', 'function': <function progress_bar>}*, *maxwarn=1*, ***mdp_options*)

Minimize a system in two steps:

1. minimisation without bond constraints
2. minimisation using bond constraint for bonds involving hydrogen

Parameters

- **out_folder** (*str*) – path of the output file folder
- **name** (*str, default=None*) – name of the simulation to run
- **no_constr_nsteps** (*int, default=1000*) – number of minimisation steps in the first step
- **constr_nsteps** (*int, default=1000*) – number of minimisation steps in the second step
- **posres** (*str, default=""*) – option for the `define` variable in the mdp file, need to be define to have postion restraints.
- **create_box_flag** (*bool, optional, default=False*) – flag to create or not a box to the input coor file.
- **monitor** (*dict, default=None*) – option to monitor a simulation, if not none monitor should contains two values: `function` the function to be ran while simulation is running and `input` parameters for the function
- **maxwarn** (*int, default=0*) – Maximum number of warnings when using `gmx grompp`
- **mdp_options** (*dict*) – Additional mdp parameters to use

Object requirement(s):

- self.coor_file
- self.top_file
- self.nt
- self.ntmpi
- self.gpu_id

Object field(s) changed:

- self.mdp
- self.tpr
- self.coor_file
- self.xtc

em_CG(*out_folder*, *name=None*, *nsteps=500000*, *maxwarn=0*, *monitor_tool={'file_check_ext': 'log', 'function': <function progress_bar>}*, ***mdp_options*)

Equilibrate a system a CG system:

1. equilibration of nsteps_HA with position restraints on Heavy Atoms with dt = dt_HA 2. equilibration of nsteps_CA with position restraints on Carbon Alpha with dt = dt 3. equilibration of nsteps_CA_LOW with position restraints on Carbon Alpha with Low restraints with dt = dt

Parameters

- **out_folder** (*str*) – path of the output file folder
- **name** (*str, default=None*) – name of the simulation to run
- **pdb_restr** (*str, default=None*) – reference coordinate file for position restraints
- **nsteps** (*int, default=100000*) – number of equilibration steps with BB constraints
- **dt** (*float, default=0.002*) – integration time step for BB equilibration
- **monitor** (*dict, default=None*) – option to monitor a simulation, if not none monitor should contains two values: `function` the function to be ran while simulation is running and `input` parameters for the function
- **mdp_options** (*dict*) – Additional mdp parameters to use

Object requirement(s):

- self.coor_file
- self.top_file
- self.nt
- self.ntmpi
- self.gpu_id

Object field(s) changed:

- self.mdp
- self.tpr
- self.coor_file
- self.xtc

em_equi_three_step_iter_error(*out_folder, name=None, no_constr_nsteps=1000, constr_nsteps=1000, pdb_restr=None, nsteps_HA=100000, nsteps_CA=200000, nsteps_CA_LOW=400000, dt=0.002, dt_HA=0.001, maxwarn=0, iter_num=3, monitor_tool={'file_check_ext': 'log', 'function': <function progress_bar>}, vsite='none', **mdp_options*)

Minimize a system in 2 steps:

1. minimisation without bond constraints
2. minimisation using bond constraint for bonds involving hydrogen

Equilibrate a system in 3 steps:

1. equilibration of nsteps_HA with position restraints on Heavy Atoms with dt = dt_HA 2. equilibration of nsteps_CA with position restraints on Carbon Alpha with dt = dt 3. equilibration of nsteps_CA_LOW with position restraints on Carbon Alpha with Low restraints with dt = dt

In case this process will crash (eg. LINCS WARNING …), the process will be rerun for *iter_num* time.

Parameters

- **out_folder** (*str*) – path of the output file folder
- **name** (*str, default=None*) – name of the simulation to run
- **no_constr_nsteps** (*int, default=1000*) – number of minimisation steps in the first step

- **constr_nsteps** (*int, default=1000*) – number of minimisation steps in the second step
- **pdb_restr** (*str, default=None*) – reference coordinate file for position restraints
- **nsteps_HA** (*int, default=100000*) – number of equilibration steps with HA constraints
- **nsteps_CA** (*int, default=200000*) – number of equilibration steps with CA constraints
- **nsteps_CA_LOW** (*int, default=400000*) – number of equilibration steps with CA_LOW constraints
- **dt_HA** (*float, default=0.001*) – integration time step for HA equilibration
- **dt** (*float, default=0.002*) – integration time step for CA and CA_LOW equilibration
- **maxwarn** (*int, default=0*) – Maximum number of warnings when using `gmx grompp`
- **monitor** (*dict, default=None*) – option to monitor a simulation, if not none monitor should contains two values: `function` the function to be ran while simulation is running and `input` parameters for the function
- **vsite** (*str, optional, default="none"*) – option for bonds constraints ("none")
- **mdp_options** (*dict*) – Additional mdp parameters to use

Object requirement(s):

- self.coor_file
- self.top_file
- self.nt
- self.ntmpi
- self.gpu_id

Object field(s) changed:

- self.mdp
- self.tpr
- self.coor_file
- self.xtc

equi_CG(*out_folder*, *name=None*, *pdb_restr=None*, *nsteps=500000*, *dt=0.02*, *maxwarn=0*, *monitor_tool={'file_check_ext': 'log', 'function': <function progress_bar>}*, ***mdp_options*)

Equilibrate a system a CG system:

1. equilibration of nsteps_HA with position restraints on Heavy Atoms with dt = dt_HA 2. equilibration of nsteps_CA with position restraints on Carbon Alpha with dt = dt 3. equilibration of nsteps_CA_LOW with position restraints on Carbon Alpha with Low restraints with dt = dt

Parameters

- **out_folder** (*str*) – path of the output file folder
- **name** (*str, default=None*) – name of the simulation to run
- **pdb_restr** (*str, default=None*) – reference coordinate file for position restraints
- **nsteps** (*int, default=100000*) – number of equilibration steps with BB constraints

- **dt** (*float, default=0.002*) – integration time step for BB equilibration
- **monitor** (*dict, default=None*) – option to monitor a simulation, if not none monitor should contains two values: `function` the function to be ran while simulation is running and `input` parameters for the function
- **mdp_options** (*dict*) – Additional mdp parameters to use

Object requirement(s):

- self.coor_file
- self.top_file
- self.nt
- self.ntmpi
- self.gpu_id

Object field(s) changed:

- self.mdp
- self.tpr
- self.coor_file
- self.xtc

equi_three_step(*out_folder, name=None, pdb_restr=None, nsteps_HA=100000, nsteps_CA=200000, nsteps_CA_LOW=400000, dt=0.002, dt_HA=0.001, maxwarn=0, monitor_tool={'file_check_ext': 'log', 'function': <function progress_bar>}, vsite='none', **mdp_options*)

Equilibrate a system in 3 steps:

1. equilibration of nsteps_HA with position restraints on Heavy Atoms with dt = dt_HA 2. equilibration of nsteps_CA with position restraints on Carbon Alpha with dt = dt 3. equilibration of nsteps_CA_LOW with position restraints on Carbon Alpha with Low restraints with dt = dt

Parameters

- **out_folder** (*str*) – path of the output file folder
- **name** (*str, default=None*) – name of the simulation to run
- **pdb_restr** (*str, default=None*) – reference coordinate file for position restraints
- **nsteps_HA** (*int, default=100000*) – number of equilibration steps with HA constraints
- **nsteps_CA** (*int, default=200000*) – number of equilibration steps with CA constraints
- **nsteps_CA_LOW** (*int, default=400000*) – number of equilibration steps with CA_LOW constraints
- **dt_HA** (*float, default=0.001*) – integration time step for HA equilibration
- **dt** (*float, default=0.002*) – integration time step for CA and CA_LOW equilibration
- **maxwarn** (*int, default=0*) – Maximum number of warnings when using `gmx grompp`
- **monitor** (*dict, default=None*) – option to monitor a simulation, if not none monitor should contains two values: `function` the function to be ran while simulation is running and `input` parameters for the function

- **vsite** (*str, optional, default="none"*) – option for bonds constraints ("none")
- **mdp_options** (*dict*) – Additional mdp parameters to use

Object requirement(s):

- self.coor_file
- self.top_file
- self.nt
- self.ntmpi
- self.gpu_id

Object field(s) changed:

- self.mdp
- self.tpr
- self.coor_file
- self.xtc

Note: In case of LINCS warning or segmentation fault, try to center the protein in the box using the center_mol_box() function.

---

extend_sim(*tpr_file=None, nsteps=200000, monitor_tool={'file_check_ext': 'log', 'function': <function progress_bar>}*)

Extend a simulation run.

Parameters

- **tpr_file** – path of the tpr file
- **nsteps** (*int, default=200000*) – number of steps
- **monitor** (*dict, default=None*) – option to monitor a simulation, if not none monitor should contains two values: `function` the function to be ran while simulation is running and `input` parameters for the function

Object requirement(s):

- self.tpr
- self.nt
- self.ntmpi
- self.gpu_id

Object field(s) changed:

- self.tpr
- self.sim_name
- self.coor_file
- self.xtc

> **Warning:** The simulation can only run nsteps more steps. Should be improved to check how many steps have been run, and give a total number of step to compute.

extract_mol_sys(*out_folder*, *res_name*)

Extract a molecule topologie and coordinates from a system:

Parameters

- **out_folder** (*str*) – path of the output file folder
- **res_name** (*str*) – Molecule residue name to extract

Returns
The GmxSys of the molecule alone

Return type
GmxSys

Note: This function does not use or affect the GmxSys object.

---

get_all_output()

In a case of a simulation restart, outputs edr, log, gro, xvg and xtc files are called for example as `self.sim_name+".partXXXX.edr"` where XXXX is the iteration number of restart (eg. first restart: XXXX=0002).

This function return a dictionnary of all edr, log, coor_file, xvg and xtc list.

Object requirement(s):

- self.sim_name

Returns
return dict containing edr, log, xtc, xvg and coor_file file list

Return type
dict

get_angle(*angle_list*, *output_xvg='tmp_angle.xvg'*, *keep_ener_file=False*, *improper=False*)

Get angle of a traj using `gmx angle`.

Parameters

- **angle_list** (*list*) – List of atom triplet
- **output_xvg** (*str, default='tmp_angle.xvg'*) – output *.xvg* file name
- **keep_ener_file** (*bool, default=False*) – flag to keep or not output *.xvg* file.
- **improper** (*bool, default=False*) – flag to compute improper angles.

Returns
Distance table

Return type
pd.DataFrame

get_dist(*distance_list*, *output_xvg='tmp_dist.xvg'*, *keep_ener_file=False*)

Get distances as a function of time for a trajectory using `gmx distance`.

Parameters

- **distance_list** (*list*) – List of atom couple
- **output_xvg** (*str, default='tmp_dist.xvg'*) – output *.xvg* file name
- **keep_ener_file** (*bool, default=False*) – flag to keep or not output *.xvg* file.

Returns
Distance table

Return type
pd.DataFrame

get_ener(*selection_list*, *output_xvg='tmp_edr.xvg'*, *check_file_out=True*, *keep_ener_file=False*)

Get energy of a system using `gmx energy`.

Parameters

- **selection_list** (*list*) – List of selection names or number
- **output_xvg** (*str, default='tmp_edr.xvg'*) – output *.xvg* file name
- **check_file_out** (*bool, default=True*) – flag to check or not if file has already been created. If the file is present then the command break.
- **keep_ener_file** (*bool, default=False*) – flag to keep or not output *.xvg* file.

Returns
Energy table

Return type
pd.DataFrame

get_index_dict()

Read and *.ndx* file and return and dictionnary with keys being the group name, nad value the index.

get_last_output()

In a case of a simulation restart, outputs edr, log, gro and xtc files are called for example as `self.sim_name+".partXXXX.edr"` where XXXX is the iteration number of restart (eg. first restart: XXXX=0002).

This function actualise the edr, log, coor_file and xtc variable.

Object requirement(s):

- self.sim_name

Object field(s) changed:

- self.edr
- self.log
- self.coor_file
- self.xtc

get_mdp_dict(*mdp_file=None*)

Extract mdp file's parameters and return it as a dict. By default read self.coor, but if mdp_file option is defined, it will read it.

Parameters
mdp_file (*str, default=None*) – mdp file

Object requirement(s):

- self.mdp

get_rmsd(*selection_list=['C-alpha', 'Protein']*, *output_xvg='tmp_rmsd.xvg'*, *fit='rot+trans'*, *pbc='no'*, *ref_sys=None*, *keep_ener_file=False*)

Get RMSD of a system using `gmx rms`.

Parameters

- **selection_list** (*list, default=['C-alpha', 'Protein']*) – List of selection names or number
- **output_xvg** (*str, default='tmp_rmsd.xvg'*) – output *.xvg* file name
- **fit** (*str, default="rot+trans"*) – Coordinates Fitting Method
- **pbc** (*str, default="no"*) – Peridic Boundary condition treatment
- **keep_ener_file** (*bool, default=False*) – flag to keep or not output *.xvg* file.
- **ref_sys** (*GmxSys, default=None*) – reference system for RMSD calculation

Returns
RMSD table

Return type
pd.DataFrame

get_rmsf(*selection_list*, *output_xvg='tmp_rmsf.xvg'*, *fit='no'*, *res='no'*, *keep_ener_file=False*)

Get RMSF of a system using `gmx rmsf`.

Parameters

- **selection_list** (*list*) – List of selection names or number
- **output_xvg** (*str, default='tmp_rmsf.xvg'*) – output *.xvg* file name
- **fit** (*str, default="no"*) – Flag for fitting before computing RMSF
- **res** (*str, default="no"*) – Residue averaging flag
- **keep_ener_file** (*bool, default=False*) – flag to keep or not output *.xvg* file.

Returns
RMSF table

Return type
pd.DataFrame

get_simulation_time()

In a case of a simulation restart simulation, one would like to know how much simulation time has already been computed to reach a certain amount of time in the restart simulation. The command will check the cpt file using `gmx check -f file.cpt`.

Object requirement(s):

- self.tpr

Object field(s) changed:

- None

Returns
return simulation time (ns)

Return type
float

insert_mol_sys(*mol_gromacs*, *mol_num*, *new_name*, *out_folder*, *check_file_out=True*)

Insert a new molecule in a system:

Insert structure and topologie of `mol_num` copy of `mol_gromacs` molecule, in the system with 6 successive steps:

1. Copy the molecule `mol_num` time.
2. **Change the chain ID of mol_gromacs to "Y", this step is necessary** for vmd to recognize the inserted mol.
3. Concat the two structure.
4. Insert the molecule in the solvant with a vmd script.
5. Update the topologie with the molecule and new water number.
6. If the charge is not null add ions to neutralize the system.

Parameters

- **mol_gromacs** (*GmxSys object*) – molecule object to be inserted
- **mol_num** (*int*) – molecule number to be inserted
- **new_name** (*str*) – generic name of the system
- **out_folder** (*str*) – path of the output file folder
- **check_file_out** (*bool, optional, default=True*) – flag to check or not if file has already been created. If the file is present then the command break.

Object requirement(s):

- self.coor_file
- self.top_file

Object field(s) changed:

- self.coor_file
- self.top_file

Note: VMD don't need anymore to be installed to run the peptide creation

---

property log

property mdp

property ndx

prepare_top(*out_folder*, *name=None*, *vsite='none'*, *ignore_ZN=True*, *ff='charmm36-jul2017'*, *ph=7.0*, *res_prot_dict=None*, *include_mol={}*)

Prepare the topologie of a protein:

1. compute hisdine protonation with `pdb2pqr`.
2. Change Histidine resname according to the protonation.
3. Correct cystein resname.
4. Correct chain ID's.

5. **Zinc Finger: Add Zinc in the pdb and change residue type of** CYS and HIS coordinating the Zinc.
6. Finally compute the topologie with pdb2gmx add_top().

Parameters

- **out_folder** (*str*) – path of the output file folder
- **name** (*str, optional, default=None*) – generic name of the system
- **vsite** (*str, optional, default="none"*) – option for topologie's bonds constraints ("none", "hydrogens", "aromatics")
- **ignore_ZN** (*bool, optional, default=False*) – option for not adding parameters to ZINC finger
- **ff** (*str, optional, default="charmm36-jul2017"*) – forcefield
- **ph** (*float, optional, default=7.0*) – pH to assign AA protonation (using pdb2pqr)
- **res_prot_dict** (*dict, optional, default=None*) – option to define manually protonation
- **include_mol** (*list, optional, default=[]*) – list of ligand's residue name to include

Object requirement(s):

- self.coor_file

Object field(s) changed:

- self.coor_file
- self.top_file

Example

```
>>> TEST_OUT = getfixture('tmpdir')
>>> # Create the topologie of a protein and do a minimisation:
>>> dna_lig = GmxSys(name='1D30', coor_file=TEST_PATH+'/1D30.pdb')
>>> dna_lig.prepare_top(out_folder=TEST_OUT+'/prepare_top/top_dna/', ff=
↪'amber99sb-ildn', include_mol={'DAP':'NC(=N)c1ccc(cc1)c2[nH]c3cc(ccc3c2)C(N)=N
↪'})
Succeed to read file .../test_files/1D30.pdb ,  532 atoms found
Succeed to read file .../test_files/1D30.pdb ,  532 atoms found
Succeed to read file .../test_files/1D30.pdb ,  532 atoms found
Succeed to save file 00_1D30.pqr
Succeed to read file 00_1D30.pqr ,  486 atoms found
Succeed to read file .../test_files/1D30.pdb ,  532 atoms found
Succeed to save file DAP.pdb
Succeed to read file DAP.pdb ,  21 atoms found
Succeed to save file DAP_0.pdb
Succeed to read file DAP_0.pdb ,  21 atoms found
Succeed to read file DAP_0_h.pdb ,  36 atoms found
Succeed to save file DAP_0_h.pdb
Succeed to read file DAP_0_h.pdb ,  36 atoms found
Succeed to save file DAP_0_h.pdb
```

(continues on next page)

(continued from previous page)

```
Succeed to save concat file:  DAP_h.pdb
Succeed to read file DAP_h.pdb ,  36 atoms found
Succeed to save file DAP_h_unique.pdb
acpype... -i DAP_h_unique.pdb -b DAP -c bcc -a gaff -o gmx -n 0
Succeed to save file 01_1D30_good_his.pdb
- Create topologie
gmx pdb2gmx -f 01_1D30_good_his.pdb -o 1D30_pdb2gmx.pdb -p 1D30_pdb2gmx.top -i␣
↪1D30_posre.itp -water tip3p -ff amber99sb-ildn -ignh -vsite none
Add Molecule: DAP
Add 1 mol .../top_dna/DAP.acpype/DAP_GMX.itp
Concat files: ['1D30_pdb2gmx.pdb', '.../top_dna/DAP_h.pdb']
Succeed to save concat file:  1D30_pdb2gmx_mol.pdb
>>> dna_lig.em(out_folder=TEST_OUT + '/prepare_top/em_dna', nsteps=10, create_
↪box_flag=True, maxwarn=1)
- Create pbc box
gmx editconf -f .../top_dna/1D30_pdb2gmx_mol.pdb -o .../top_dna/1D30_pdb2gmx_
↪mol_box.pdb -bt dodecahedron -d 1.0
- Create the tpr file 1D30.tpr
gmx grompp -f 1D30.mdp -c .../top_dna/1D30_pdb2gmx_mol_box.pdb -r .../top_dna/
↪1D30_pdb2gmx_mol_box.pdb -p .../top_dna/1D30_pdb2gmx_mol.top -po out_1D30.mdp␣
↪-o 1D30.tpr -maxwarn 1
- Launch the simulation 1D30.tpr
gmx mdrun -s 1D30.tpr -deffnm 1D30 -nt 0 -ntmpi 0 -nsteps -2 -nocopyright
>>> lig = dna_lig.extract_mol_sys(out_folder=TEST_OUT+'/prepare_top/top_DAP/',␣
↪res_name='DAP')
- Convert trj/coor
gmx trjconv -f ...1D30.gro -o ...1D30_compact.pdb -s ...1D30.tpr -ur compact -
↪pbc mol
Succeed to read file ...1D30_compact.pdb ,  794 atoms found
Succeed to save file ...DAP_only.pdb
Forcefield include :
 amber99sb-ildn
- ITP file: DAP_GMX_atomtypes
- molecules defined in the itp file:
- ITP file: tip3p
- molecules defined in the itp file:
* SOL
- ITP file: DAP_GMX
- molecules defined in the itp file:
* DAP
Mol List:
   * 1 DAP
Mol Name:
 Protein
>>> lig.create_box()
- Create pbc box
gmx editconf -f ...DAP_only.pdb -o ...DAP_only_box.pdb -bt dodecahedron -d 1.0
>>> lig.solvate_box(out_folder=TEST_OUT + '/prepare_top/water_lig_top')
- Solvate the pbc box
Copy topologie file and dependancies
>>> lig.em(out_folder=TEST_OUT + '/prepare_top/em_water_DAP', nsteps=10,␣
↪maxwarn=1)
```

(continues on next page)

(continued from previous page)

```
- Create the tpr file DAP.tpr
gmx grompp -f DAP.mdp -c ...DAP_water.pdb -r ...DAP_water.pdb -p ...DAP_water.
↪top -po out_DAP.mdp -o DAP.tpr -maxwarn 1
- Launch the simulation DAP.tpr
gmx mdrun -s DAP.tpr -deffnm DAP -nt 0 -ntmpi 0 -nsteps -2 -nocopyright
```

Note: No options are allowed (forcefield, water model, termini capping) except for vsites.

prepare_top_ligand(*out_folder*, *name=None*, *ff='amber99sb-ildn'*, *water_model='tip3p'*, *include_mol={}*)

Prepare the topologie of a ligand:

Parameters

- **out_folder** (*str*) – path of the output file folder
- **name** (*str, optional, default=None*) – generic name of the system
- **ff** (*str, optional, default="amber99sb-ildn"*) – forcefield
- **include_mol** (*list, optional, default=[]*) – list of ligand's residue name to include

Object requirement(s):

- self.coor_file

Object field(s) changed:

- self.coor_file
- self.top_file

Example

```
>>> TEST_OUT = getfixture('tmpdir')
>>> # Create the topologie of a protein and do a minimisation:
>>> lig = GmxSys(name='1D30', coor_file=TEST_PATH+'/1D30.pdb')
```

Note: Starting file need to be a pdb, this should be changed.

prod_CG(*out_folder*, *name=None*, *nsteps=5000000*, *dt=0.02*, *maxwarn=0*, *monitor_tool={'file_check_ext': 'log', 'function': <function progress_bar>}*, ***mdp_options*)

Equilibrate a system a CG system:

1. equilibration of nsteps_HA with position restraints on Heavy Atoms with dt = dt_HA 2. equilibration of nsteps_CA with position restraints on Carbon Alpha with dt = dt 3. equilibration of nsteps_CA_LOW with position restraints on Carbon Alpha with Low restraints with dt = dt

Parameters

- **out_folder** (*str*) – path of the output file folder
- **name** (*str, default=None*) – name of the simulation to run
- **nsteps** (*int, default=100000*) – number of equilibration steps with BB constraints
- **dt** (*float, default=0.002*) – integration time step for BB equilibration

- **monitor** (*dict, default=None*) – option to monitor a simulation, if not none monitor should contains two values: `function` the function to be ran while simulation is running and `input` parameters for the function
- **mdp_options** (*dict*) – Additional mdp parameters to use

Object requirement(s):

- self.coor_file
- self.top_file
- self.nt
- self.ntmpi
- self.gpu_id

Object field(s) changed:

- self.mdp
- self.tpr
- self.coor_file
- self.xtc

production(*out_folder*, *name=None*, *nsteps=400000*, *dt=0.002*, *maxwarn=0*, *monitor_tool={'file_check_ext': 'log', 'function': <function progress_bar>}*, *vsite='none'*, ***mdp_options*)

Run a production run.

Parameters

- **out_folder** (*str*) – path of the output file folder
- **name** (*str, default=None*) – name of the simulation to run
- **nsteps** (*int, default=400000*) – number of minimisation steps
- **dt** (*float, default=0.002*) – number of minimisation steps
- **maxwarn** (*int, default=0*) – Maximum number of warnings when using `gmx grompp`
- **monitor** (*dict, default=None*) – option to monitor a simulation, if not none monitor should contains two values: `function` the function to be ran while simulation is running and `input` parameters for the function
- **vsite** (*str, optional, default="none"*) – option for topologie's bonds constraints ("none", "hydrogens", "all")
- **mdp_options** (*dict*) – Additional mdp parameters to use

Object requirement(s):

- self.coor_file
- self.top_file
- self.nt
- self.ntmpi
- self.gpu_id

Object field(s) changed:

- self.mdp

- self.tpr
- self.coor_file
- self.xtc

run_md_sim(*out_folder*, *name*, *mdp_template*, *mdp_options*, *pdb_restr=None*, *maxwarn=0*, *monitor_tool={'file_check_ext': 'log', 'function': <function progress_bar>}*)

Run a simulation using 3 steps:

1. Create a mdp file
2. Create a tpr file using `gmx grompp`
3. Launch the simulation using `gmx mdrun`

Parameters

- **out_folder** (*str*) – path of the output file folder
- **name** (*str*) – name of the simulation to run
- **mdp_template** (*str*) – mdp file template
- **mdp_options** (*dict*) – New parameters to use
- **pdb_restr** (*str, default=None*) – reference coordinate file for position restraints
- **maxwarn** (*int, default=0*) – Maximum number of warnings when using `gmx grompp`
- **monitor** (*dict, default=None*) – option to monitor a simulation, if not none monitor should contains two values: `function` the function to be ran while simulation is running and `input` parameters for the function

Object requirement(s):

- self.coor_file
- self.top_file
- self.nt
- self.ntmpi
- self.gpu_id

Object field(s) changed:

- self.mdp
- self.tpr
- self.coor_file
- self.xtc

run_simulation(*check_file_out=True*, *cpi=None*, *nsteps=-2*, *rerun=False*, *monitor_tool={'file_check_ext': 'log', 'function': <function progress_bar>}*)

Launch the simulation using `gmx mdrun`

Parameters

- **check_file_out** (*bool, optional, default=True*) – flag to check or not if file has already been created. If the file is present then the command break.
- **cpi** (*str, default=None*) – checkpoint file, if defined, it will restart a simulation and run `nsteps`.

- **nsteps** (*int, default=-2*) – Number of steps to run, (-2 : will use mdp parameter)
- **rerun** (*bool, default=False*) – option to rerun a simulation (eg. recompute energy)
- **monitor_tool** (*dict, default=None*) – option to monitor a simulation, if not *None*, monitor should contains two values: `function` the function to be ran while simulation is running and `input` parameters for the function.

Object requirement(s):

- self.tpr
- self.sim_name
- self.nt
- self.ntmpi
- self.gpu_id

Additional requirement(s) for rerun:

- self.xtc

Object field(s) changed:

- self.coor_file
- self.xtc
- self.edr
- self.log

Note: The function must be launched in the path where the tpr is present.

Note: If cpi file is defined the simulation will restart with the `-noappend` option, if cpi is not defined, but the .cpt file exist, it will restart with "append".

---

save_state()

Save last state

static set_coor_aa_prot(*coor_in*, *res_prot_dict*, *ff*)

Set manually residue protonation.

Parameters

- **coor_in** (*Coor*) – coordinate to update
- **res_prot_dict** (*dict*) – Dictionary of protonated residues
- **ff** (*str*) – forcefield

solvate_add_ions(*out_folder*, *name=None*, *ion_C=0.15*, *create_box_flag=True*, *box_dist=1.1*, *radius=0.25*, *maxwarn=1*)

Solvate a system with three succesive steps:

1. Create box using `create_box()`
2. Add water using `solvate_box()`
3. Add ions using `add_ions()`

Parameters

- **out_folder** (*str*) – path of the output file folder
- **name** (*str, optional, default=None*) – generic name of the system
- **ion_C** (*float, optional, default=0.15*) – ionic concentraton (Molar)

Object requirement(s):

- self.coor_file
- self.top_file

Object field(s) changed:

- self.coor_file solvate_box
- self.top_file

Example

```
>>> TEST_OUT = getfixture('tmpdir')
>>> prot = GmxSys(name='1y0m', coor_file=TEST_PATH+'/1y0m.pdb')
>>> prot.add_top(out_folder=TEST_OUT+'/solvate_add_ions/top_SH3/')
- Create topologie
gmx pdb2gmx -f .../test_files/1y0m.pdb -o 1y0m_pdb2gmx.pdb -p 1y0m_pdb2gmx.top -
↪i 1y0m_posre.itp -water tip3p -ff charmm36-jul2017
Molecule topologie present in 1y0m_pdb2gmx.top , extract the topologie in a␣
↪separate file: 1y0m_pdb2gmx.itp
Protein_chain_A
- ITP file: 1y0m_pdb2gmx.itp
- molecules defined in the itp file:
* Protein_chain_A
Rewrite topologie: 1y0m_pdb2gmx.top
>>> prot.solvate_add_ions(out_folder=TEST_OUT+'/solvate_add_ions/top_SH3_water_
↪ions/')
- Create pbc box
gmx editconf -f .../solvate_add_ions/top_SH3/1y0m_pdb2gmx.pdb -o .../solvate_
↪add_ions/top_SH3/1y0m_pdb2gmx_box.pdb -bt dodecahedron -d 1.1
- Solvate the pbc box
Copy topologie file and dependancies
Copy topologie file and dependancies
- Create the tpr file genion_1y0m_water_ion.tpr
gmx grompp -f .../template/mini.mdp -c 1y0m_water.pdb -r 1y0m_water.pdb -p 1y0m_
↪water_ion.top -po out_mini.mdp -o genion_1y0m_water_ion.tpr -maxwarn 1
- Add ions to the system with an ionic concentration of 0.15 M , sytem charge =␣
↪0.0 water num= 62...
Add ions : NA : 17   CL : 17
gmx genion -s genion_1y0m_water_ion.tpr -p 1y0m_water_ion.top -o 1y0m_water_ion.
↪gro -np 17 -pname NA -nn 17 -nname CL
>>> prot.em(out_folder=TEST_OUT+'/solvate_add_ions/em_SH3_water_ions/',␣
↪nsteps=10, constraints = "none")
- Create the tpr file 1y0m.tpr
gmx grompp -f 1y0m.mdp -c ../top_SH3_water_ions/1y0m_water_ion.gro -r ../top_
↪SH3_water_ions/1y0m_water_ion.gro -p ../top_SH3_water_ions/1y0m_water_ion.top␣
↪-po out_1y0m.mdp -o 1y0m.tpr -maxwarn 1
```

(continues on next page)

(continued from previous page)

```
- Launch the simulation 1y0m.tpr
gmx mdrun -s 1y0m.tpr -deffnm 1y0m -nt 0 -ntmpi 0 -nsteps -2 -nocopyright
```

Note: If name is not defined, it will use the object name.

solvate_box(*out_folder*, *name=None*, *radius=0.21*, *cs='share/gromacs/top/spc216.gro'*, *check_file_out=True*)

Solvate the pbc box with water or another mol defined with `cs` using the `gmx solvate` command.

Parameters

- **out_folder** (*str*) – path of the output file folder
- **name** (*float, optional, default=0.21*) – generic name of the system
- **radius** – default van der Waals distance (nm)
- **cs** (*str, optional, default=``WATER_GRO``*) – solvant coordinate file
- **check_file_out** (*bool, optional, default=True*) – flag to check or not if file has already been created. If the file is present then the command break.

Object requirement(s):

- self.coor_file
- self.top_file

Object field(s) changed:

- self.coor_file
- self.top_file

Example

```
>>> TEST_OUT = getfixture('tmpdir')
>>> prot = GmxSys(name='1y0m', coor_file=TEST_PATH+'/1y0m.pdb')
>>> prot.add_top(out_folder=TEST_OUT+'/solv_box/top_SH3/')
- Create topologie
gmx pdb2gmx -f .../test_files/1y0m.pdb -o 1y0m_pdb2gmx.pdb -p 1y0m_pdb2gmx.top -
↪i 1y0m_posre.itp -water tip3p -ff charmm36-jul2017
Molecule topologie present in 1y0m_pdb2gmx.top , extract the topologie in a␣
↪separate file: 1y0m_pdb2gmx.itp
Protein_chain_A
- ITP file: 1y0m_pdb2gmx.itp
- molecules defined in the itp file:
* Protein_chain_A
Rewrite topologie: 1y0m_pdb2gmx.top
>>> prot.create_box()
- Create pbc box
gmx editconf -f .../solv_box/top_SH3/1y0m_pdb2gmx.pdb -o .../solv_box/top_SH3/
↪1y0m_pdb2gmx_box.pdb -bt dodecahedron -d 1.0
>>> prot.solvate_box(out_folder=TEST_OUT+'/solv_box/top_SH3_water/')
- Solvate the pbc box
Copy topologie file and dependancies
```

Note: If `name` is not defined, the command will create a new .pdb and .top file name after the object name and adding "_water".

switch_ion_octa_dummy(*ion_name=['MN', 'ZN']*)

property top_file

property tpr

view_coor()

Return a *nglview* object to view the object coordinates in a jupyter notebook with the module `nglview`.

Example

```
>>> import nglview as nv
>>> prot = GmxSys(name='1y0m', coor_file=TEST_PATH+'/1y0m.pdb')
>>> view = prot.view_coor()
>>> view
```

view_traj()

Return a *nglview* object to view the object trajectorie in a jupyter notebook with the module `nglview`.

Example

```
>>> import nglview as nv
>>> prot = GmxSys(name='1y0m', coor_file=TEST_PATH+'/1y0m.pdb')
>>> view = prot.view_traj()
>>> view
```

Note: This function has a dependencies with the `simpletraj` a lightweight coordinate-only trajectory reader. Install it using `pip install simpletraj` or `conda install simpletraj`.

property xtc

property xvg

gromacs_py.gmx.gmxsys.**show_debug**(*pdb_manip_log=True*)

To use only with Doctest !!! Redirect logger output to sys.stdout

gromacs_py.gmx.gmxsys.**show_log**(*pdb_manip_log=True*)

To use only with Doctest !!! Redirect logger output to sys.stdout

gromacs_py.gmx.itp module

class gromacs_py.gmx.itp.**Itp**(*name*, *fullname*, *path*)

Bases: `object`

Itp topologie in gromacs format May contain several molecule definition, so itp class is a collection of top_mol object which are indivudial molecule topologies

add_posre(*mol_name*, *posre_name*, *selec_dict*, *fc*, *replace=True*)

change_mol_name(*old_name*, *new_name*)

charge(*mol_name*)

display()

get_include_file_list()

read_file()

res_num(*mol_name*)

set_top_mol_name(*new_name*)

write_file(*itp_file*)

gromacs_py.gmx.itp.**show_log**()

To use only with Doctest !!! Redirect logger output to sys.stdout

gromacs_py.gmx.itp.**write_index_posre_file**(*atom_index_list*, *posre_file*, *type_val=1*, *fc=[1000, 1000, 1000]*)

Write a pos restraint file based on atom index list

gromacs_py.gmx.rtp module

class gromacs_py.gmx.rtp.**Rtp**(*path*)

Bases: `object`

Individual molecule topologie

read_file()

gromacs_py.gmx.topmol module

class gromacs_py.gmx.topmol.**TopMol**(*name*, *nrexcl*)

Bases: `object`

Individual molecule topologie

add_atoms(*index*, *atom_list*)

Add a list of atoms in atom_dict at an index position. Correct all indexes in bond, pairs, …

Parameters

- **index** (*int*) – index for insertion
- **atom_list** (*list*) – list of atoms to add

correct_charge_type(*forcefield*, *index_list=None*)

Correct the charge and atom type of an itp object, base on a ff *.rtp* file. This is specially usefull, to correct charge of termini resiudes of a cyclicized peptide.

if index_list is None, will correct all atom charges, if not, only atoms listed in index_list.

delete_atom(*index_list*)

get_charge()

get_res_num()

get_selection_index(*selec_dict={'atom_name': ['CA']}*)

Return the atom index to add posre

write_file(*filout*)

gromacs_py.gmx.topmol.**show_debug**()

To use only with Doctest !!! Redirect logger output to sys.stdout

gromacs_py.gmx.topmol.**show_log**()

To use only with Doctest !!! Redirect logger output to sys.stdout

gromacs_py.gmx.topsys module

class gromacs_py.gmx.topsys.**TopSys**(*top_in*)

Bases: object

Topologie base on gromacs .top : #include forcefield

All name and full path of itp are save [system] -> Name [molecules] -> Composition

Parameters

- **path** (*str*) – topologie file path
- **forcefield** (*str*) – name of the focefield
- **itp_list** (*list*) – list of the itp object
- **mol_comp** (*list*) – molecular composition
- **name** – name of the system
- **folder** (*str*) – path of the top file folder
- **include_itp** (*bool*) – Flag indicating if the topologie include a molecule topologie

add_atomtypes(*new_atomtypes*)

Add atomtypes in a topologie.

Parameters

new_atomtypes (*str*) – path of the atomtype itp file

add_atomtypes_2(*atomtypes_dict*)

Add atomtypes in a topologie.

Parameters

atomtypes_dict (*str*) – atom types dict

add_intermolecular_restr(*bond_list=[], angle_list=[], dihed_list=[]*)

Add inter molecular restraints in topologie file

add_mol(*mol_name, mol_itp_file, mol_num*)

Add a molecule in the topologie (composition and itp_list)

add_mol_itp(*mol_itp_file*)

Add a molecule itp in the topologie itp_list.

add_posre(*posre_name='POSRE_CA', selec_dict={'atom_name': ['CA']}, fc=[1000, 1000, 1000]*)

Add position restraint based on the selection for each itp

change_mol_name(*old_name*, *new_name*)

change_mol_num(*mol_name*, *mol_num*)

Update molecule number. And remove multiple molecule definition if they are consecutive.

charge()

Get the charge of the system

copy_dependancies(*dest_folder*)

copy_top_and_dependancies(*dest_file*)

display()

get_include_file_list()

get_include_no_posre_file_list()

mol_num(*name*)

Get the number of the molecule "name"

prot_res_num(*selection='Protein'*)

Compute the residue number of a selection

read_file(*top_in*)

remove_ion(*ion_name_list*)

Remove a molecule from the topologie (composition and itp_list)

write_file(*top_out*)

gromacs_py.gmx.topsys.**show_debug**(*pdb_manip_log=True*)

To use only with Doctest !!! Redirect logger output to sys.stdout

gromacs_py.gmx.topsys.**show_log**()

To use only with Doctest !!! Redirect logger output to sys.stdout

Module contents

gmx library include the gromacs system class `GmxSys`, as well as topologie `TopSys` and itp.

gromacs_py.gmx.**show_debug**(*pdb_manip_log=True*)

gromacs_py.gmx.**show_log**(*pdb_manip_log=True*)

gromacs_py.test package

Submodules

gromacs_py.test.datafiles module

Location of datafiles for gromacs_py tests

Use it as:

` from gromacs_py.test.datafiles import * `

gromacs_py.test.test_FreeEner module

gromacs_py.test.test_GmxSys module

Module contents

Unit test package for gromacs_py.

gromacs_py.tools package

Submodules

gromacs_py.tools.monitor module

Collection of function to monitor a simulation in real time.

`gromacs_py.tools.monitor.`**`extract_log_dict`**(*func_input_dict*, *tail_line_num=20*)

Read the last lines of a gromacs `.log` file and return a dictionnary containing `time`, `step` and all energetic terms.

`gromacs_py.tools.monitor.`**`isnotebook`**()

Return if the command is launch from a notebook or not Taken from: https://stackoverflow.com/questions/15411967/ how-can-i-check-if-code-is-executed-in-the-ipython-notebook

Example

```
>>> isnotebook()
False
```

`gromacs_py.tools.monitor.`**`print_log_file`**(*proc*, *func_input_dict*, *tail_line_num=20*)

Monitor `.log` file information. The `func_input_dict` should contains several keys:

- ***terms*: list of energetic terms to extract, eg. ['Potential',** 'Temperature']
- *log*: path of the log file (Defined in `os_command.run_background()`)
- *refresh_time*: time interval to refresh log extract (default=1.0 s)

Parameters

- **proc** (*subprocess object*) – running subprocess
- **func_input_dict** (*dict*) – dictionnary containing parameters for log extract
- **tail_line_num** (*int, default=20*) – number of line to read at the end of `.log` file

Example:

```
>>> TEST_OUT = str(getfixture('tmpdir'))
>>> import sys
>>> sys.path.insert(0, os.path.abspath(os.path.join(MONITOR_LIB_DIR, '../..')))
>>> from gromacs_py import gmx

...
>>> prot = gmx.GmxSys(name='1y0m', coor_file=TEST_PATH+'/1y0m.pdb')
```

(continues on next page)

(continued from previous page)

```
>>> ##################################
>>> ####   Create the topologie:   ###
>>> ##################################
>>> prot.prepare_top(out_folder=os.path.join(TEST_OUT, 'top_SH3'), vsite='hydrogens
↪')
pdb2pqr30... --ff CHARMM --ffout CHARMM --keep-chain --titration-state-
↪method=propka --with-ph=7.00 tmp_pdb2pqr.pdb 00_1y0m.pqr
gmx pdb2gmx -f 01_1y0m_good_his.pdb -o 1y0m_pdb2gmx.pdb -p 1y0m_pdb2gmx.top -i 1y0m_
↪posre.itp -water tip3p -ff charmm36-jul2017 -ignh -vsite hydrogens
>>> #####################################
>>> ### Monitor an energy minimisation ###
>>> #####################################
>>> monitor = {'function': print_log_file,           'terms':['Potential'],        ␣
↪  'file_check_ext':'log'}
>>> prot.em(out_folder=os.path.join(TEST_OUT, 'em_SH3'), nsteps=50,    constraints=
↪'none', create_box_flag=True, monitor=monitor, nstlog=1)
gmx editconf -f .../top_SH3/1y0m_pdb2gmx.pdb -o .../top_SH3/1y0m_pdb2gmx_box.pdb -
↪bt dodecahedron -d 1.0
gmx grompp -f 1y0m.mdp -c ../top_SH3/1y0m_pdb2gmx_box.pdb -r ../top_SH3/1y0m_
↪pdb2gmx_box.pdb -p ../top_SH3/1y0m_pdb2gmx.top -po out_1y0m.mdp -o 1y0m.tpr -
↪maxwarn 1
gmx mdrun -s 1y0m.tpr -deffnm 1y0m -nt 0 -ntmpi 0 -nsteps -2 -nocopyright...
```

`gromacs_py.tools.monitor.progress_bar`(*proc*, *func_input_dict*, *tail_line_num=20*)

Monitor `.log` file timestep. The `func_input_dict` should contains several keys:

- *nsteps*: Total number of steps during the simulation
- *log*: path of the log file (Defined in `os_command.run_background()`)
- *refresh_time*: time interval to refresh log extract (default=1.0 s)

Parameters

- **proc** (*subprocess object*) – running subprocess
- **func_input_dict** (*dict*) – dictionnary containing parameters for log extract
- **tail_line_num** (*int, default=20*) – number of line to read at the end of `.log` file

Example:

```
>>> TEST_OUT = str(getfixture('tmpdir'))
>>> import sys
>>> sys.path.insert(0, os.path.abspath(os.path.join(MONITOR_LIB_DIR, '../..')))
>>> from gromacs_py import gmx

...
>>> prot = gmx.GmxSys(name='1y0m', coor_file=TEST_PATH+'/1y0m.pdb')
>>> ##################################
>>> ####   Create the topologie:   ###
>>> ##################################
>>> prot.prepare_top(out_folder=os.path.join(TEST_OUT, 'top_SH3'), vsite='hydrogens
↪')
pdb2pqr30... --ff CHARMM --ffout CHARMM --keep-chain --titration-state-
↪method=propka --with-ph=7.00 tmp_pdb2pqr.pdb 00_1y0m.pqr
```

(continues on next page)

(continued from previous page)

```
gmx pdb2gmx -f 01_1y0m_good_his.pdb -o 1y0m_pdb2gmx.pdb -p 1y0m_pdb2gmx.top -i 1y0m_
↪posre.itp -water tip3p -ff charmm36-jul2017 -ignh -vsite hydrogens
>>> ####################################
>>> ### Monitor an energy minimisation ###
>>> ####################################
>>> monitor = PROGRESS_BAR
>>> prot.em(out_folder=os.path.join(TEST_OUT, 'em_SH3'), nsteps=50,    constraints=
↪'none', create_box_flag=True, monitor=monitor, nstlog=1)
gmx editconf -f .../top_SH3/1y0m_pdb2gmx.pdb -o .../top_SH3/1y0m_pdb2gmx_box.pdb -
↪bt dodecahedron -d 1.0
gmx grompp -f 1y0m.mdp -c ../top_SH3/1y0m_pdb2gmx_box.pdb -r ../top_SH3/1y0m_
↪pdb2gmx_box.pdb -p ../top_SH3/1y0m_pdb2gmx.top -po out_1y0m.mdp -o 1y0m.tpr -
↪maxwarn 1
gmx mdrun -s 1y0m.tpr -deffnm 1y0m -nt 0 -ntmpi 0 -nsteps -2 -nocopyright
```

`gromacs_py.tools.monitor.`**`read_xvg`**(*xvg_file*)

Read a *.xvg* file and return a pandas dataframe.

Parameters

- **xvg_file** (*string (Default: 'time')*) – path of the xvg file
- **x_axis** – name of first column

Example

```
>>> xvg_file = os.path.join(TEST_PATH, 'volume.xvg')
>>> vol_df = read_xvg(xvg_file)
>>> vol_df.head()
   Time (ps)      Volume
0        0.0  171.237213
1        5.0  135.081039
2       10.0   94.224792
3       15.0   59.942383
4       20.0   58.125397
```

`gromacs_py.tools.monitor.`**`simulation_plot`**(*proc*, *func_input_dict*, *refresh_time=1.0*)

This function is used for monitoring a simulation in real time. Function can be excecuted by the gromacs.tools.os_command.run_background() function. The function monitors a trajectory file, and launch the analysis if the file has been modified. It can plot as function of time an analysis of a simulation. Analysis is passed as input function.

> **Warning:** Need to add the following lines to be run in jupyter notebook:
>
> - `%matplotlib notebook`

Example:

```
>>> TEST_OUT = str(getfixture('tmpdir'))
>>> import sys
>>> # print(os.path.abspath(os.path.join(MONITOR_LIB_DIR, '../..')))
>>> sys.path.insert(0, os.path.abspath(os.path.join(MONITOR_LIB_DIR, '../..')))
>>> from gromacs_py import gmx
```

(continues on next page)

(continued from previous page)

```
>>> prot = gmx.GmxSys(name='1y0m', coor_file=TEST_PATH+'/1y0m.pdb')
>>> ##################################
>>> ####   Create the topologie:   ###
>>> ##################################
>>> prot.prepare_top(out_folder=os.path.join(TEST_OUT, 'top_SH3'), vsite='hydrogens
↪')
pdb2pqr30... --ff CHARMM --ffout CHARMM --keep-chain --titration-state-
↪method=propka --with-ph=7.00 tmp_pdb2pqr.pdb 00_1y0m.pqr
gmx pdb2gmx -f 01_1y0m_good_his.pdb -o 1y0m_pdb2gmx.pdb -p 1y0m_pdb2gmx.top -i 1y0m_
↪posre.itp -water tip3p -ff charmm36-jul2017 -ignh -vsite hydrogens
>>> #####################################
>>> ### Monitor an energy minimisation ###
>>> #####################################
>>> monitor = {'function': simulation_plot,           'extract_func': [{'func':␣
↪extract_log_dict,                            'term': 'Potential'},              ␣
↪            {'func': extract_log_dict,                                'term':
↪'Temperature'}],           'file_check_ext':'log'}
>>> prot.em(out_folder=os.path.join(TEST_OUT, 'em_SH3'), nsteps=50,    constraints=
↪'none', create_box_flag=True, monitor=monitor, nstlog=10)
gmx editconf -f ...top_SH3/1y0m_pdb2gmx.pdb -o ...top_SH3/1y0m_pdb2gmx_box.pdb -bt␣
↪dodecahedron -d 1.0
gmx grompp -f 1y0m.mdp -c .../top_SH3/1y0m_pdb2gmx_box.pdb -r .../top_SH3/1y0m_
↪pdb2gmx_box.pdb -p .../top_SH3/1y0m_pdb2gmx.top -po out_1y0m.mdp -o 1y0m.tpr -
↪maxwarn 1
gmx mdrun -s 1y0m.tpr -deffnm 1y0m -nt 0 -ntmpi 0 -nsteps -2 -nocopyright
```

Module contents

13.1.2 Submodules

13.1.3 gromacs_py.free_ener module

class gromacs_py.free_ener.**FreeEner**(*mol_name*, *out_folder*, *unit='kcal'*)

Bases: object

Free Energy encapsulation class.

This class can be used to launch and analyze free energy calculations using Free Energy Perturbation.

Parameters

- **out_folder** (*str*) – output folder
- **lambda_coul** (*list of float*) – lambda points for Coulomb
- **lambda_vdw** (*list of float*) – lambda points for Lennard Jones
- **lambda_restr** (*list of float*) – lambda points for restraints
- **xvg_file_list** (*list of string*) – List of free energy xvg files
- **lambda_sys_list** (*list of string*) – List of lambda GmxSys
- **temp** (*float*) – Temperature
- **smile** (*str*) – Ligand SMILE

add_intermol_restr_index(*rec_index_list*, *lig_index_list*, *ref_coor*, *k=41.84*, *temp=300*)

Compute and add the intermolecular restraints based on the ref_coor distance and angles.

Give three atoms for each receptor and ligand index list: R_0, R_1, R_2 and L_0, L_1, L_2 Will define:

- 1 bond:
  - $R_0 - L_0$
- 2 angles:
  - $R_0 - L_0 - L_1$
  - $R_1 - R_0 - L_0$
- 2 dihedral angles:
  - $R_0 - L_0 - L_1 - L_2$
  - $R_2 - R_1 - R_0 - L_0$

Parameters

- **rec_index_list** (*list*) – List of the receptor atom index
- **lig_index_list** (*list*) – List of the ligand atom index
- **ref_coor** (*str*) – Reference coordinates file
- **k** (*float*) – intermolecular force constant, (default= $41.84\ kcal\,mol^{-1}\,nm^{-2}$)
- **temp** (*float*) – Temperature defalult=300 K

Object requirement(s):

- self.gmxsys.coor_file
- self.gmxsys.top_file

Object field(s) changed:

- self.gmxsys.top_file

align_ref_traj(*rec_group='Protein'*)

compute_add_intermol_from_traj(*ref_coor=None*, *rec_group='Protein'*, *k=41.84*, *cutoff_prot=6.0*)

Compute intermolecular restraint from the last GmxSys trajectory. Get a coor object Get distance and angles

compute_convergence_alchemlyb(*dt=10*)

compute_convergence_gbar(*dt=10*)

static compute_lambda_point(*gmx_sys*, *lambda_iter*, *mol_name*, *out_folder*, *free_ener_option*, *pbar*, *mbar*, *em_steps*, *nvt_steps*, *npt_steps*, *prod_steps*, *maxwarn=1*, *monitor_tool={'file_check_ext': 'log', 'function': <function progress_bar>}*)

Run the different steps of a single lambda point.

Parameters

- **gmx_sys** (GmxSys) – Gmx System to start from
- **lambda_iter** (*int*) – lambda point
- **mol_name** (*str*) – Molecule residue name

- **out_folder** (*str*) – path of the output folder
- **free_ener_option** (*dict*) – Mdp options
- **pbar** (*tqmd bar*) – progress bar object
- **mbar** (*bool*) – MBAR flag
- **em_steps** (*int*) – number of minimisation steps
- **nvt_steps** (*int*) – number of NVT equilibration steps
- **npt_steps** (*int*) – number of NPT equilibration steps
- **prod_steps** (*int*) – number of production steps
- **maxwarn** (*int, default=0*) – Maximum number of warnings when using `gmx grompp`
- **monitor** – option to monitor a simulation, if not none monitor should contains two values: `function` the function to be ran while simulation is running and `input` parameters for the function

property conv_fac

Conversion factor from kT to self.unit

equilibrate_complex(*em_steps=10000*, *HA_time=0.25*, *CA_time=0.5*, *CA_LOW_time=1.0*, *dt=0.002*, *dt_HA=0.001*, *temp=300*, *receptor_grp='Protein'*, *short_steps=50000*)

Equilibrate a receptor-ligand complex system.

Parameters

- **em_steps** (*int*) – Energy minimisation step number, default=10000
- **HA_time** (*float, default=0.25*) – Heavy atoms restraints equilibration time (ns)
- **CA_time** (*float, default=0.5*) – Alpha carbon atoms restraints equilibration time (ns)
- **CA_LOW_time** (*float, default=1.0*) – Low alpha carbon atoms restraints equilibration time (ns)
- **dt** (*float, default=0.002 ps*) – Integration time step (ps)
- **dt_HA** (*float, default=0.001 ps*) – Integration time step (ps)
- **temp** (*float, default=300.0 K*) – Temperature of equilibration (K)
- **receptor_grp** (*str, default='Protein'*) – Receptor group (for temperature coupling)
- **short_steps** (*int, default=50000*) – Short equilibration steps

equilibrate_solvent_box(*em_steps=10000*, *dt=0.002*, *prod_time=10.0*, *short_steps=50000*, *temp=300.0*)

Equilibrate a solvent (water, octanol) box.

Parameters

- **em_steps** (*int, default=10000*) – Energy minimisation step number
- **dt** (*float, default=0.002 ps*) – Integration time step (ps)
- **prod_time** (*float, default=10.0 ns*) – Equilibration time (ns)
- **short_steps** (*int, default=50000*) – Short equilibration steps
- **temp** (*float, default=300.0*) – Temperature of equilibration

extend_lambda_prod(*prod_time*)

Extend free energy production.

Parameters

prod_time (*float*) – production time (ns)

static get_bar(*xvg_file_list*, *bar_xvg='bar.xvg'*, *barint_xvg='barint.xvg'*, *hist_xvg='histogram.xvg'*, *begin_time=0*, *end_time=- 1*, *check_file_out=True*, *keep_ener_file=False*)

Get energy of a system using `gmx bar`.

I don't know how to compute std like in gmx bar.

static get_conv_fac(*unit*, *temp*)

Conversion factor from kT to self.unit

get_free_ener(*begin_time=0*, *end_time=- 1*, *unit=None*)

Show free energy calculation output

NEED TO FIX STD !!

get_ligand_atoms(*ref_coor*)

get_protein_atoms_from_res(*resid*, *rec_group='Protein'*)

get_protein_atoms_from_rmsf(*ref_coor*, *lig_atom_list*, *rec_group='Protein'*, *cutoff_max=6.0*)

get_water_restr(*temp=300*)

Compute ligand restaint energy in water using Boresh et al. equation:

$$\Delta G_{restr} = RT \ln \left(\frac{8\pi^2 V^0}{r_0^2 \sin\theta_{a0} \sin\theta_{b0}} \frac{\sqrt{k_r k_{\theta_a} k_{\theta_b} k_{\tau_\alpha} k_{\tau_\beta} k_{\tau_\gamma}}}{2\pi K T^3} \right)$$

octanol_box_from_SMILE(*smile*, *method_3d='rdkit'*, *iter_num=5000*, *box_dist=1.3*)

Create an octanol box coordinates and toplogie with a molecule defined by its SMILE.

Parameters

- **smile** (*str*) – Molecule's SMILE
- **method_3d** (*str, default='rdkit'*) – Method to compute 3D coordinates
- **iter_num** (*int, default=5000*) – Iteration step number for 3D coordinate computation
- **box_dist** (*float, default=1.3 nm*) – Ditance to egde box (nm)

Note: Default box dist 1.3 nm was taken as minimal distance for CH4 molecule, to allow domain decomposition with *gmx mdrun*.

---

plot_convergence(*graph_out=None*, *dt=10*)

plot_convergence_graph(*graph_out=None*)

plot_intermol_restr(*graph_out=None*)

prepare_complex_pdb(*pdb_in*, *smile*, *ff='amber99sb-ildn'*)

Prepare topologie from a pdb file, create box around and solvate it.

Parameters

- **pdb_in** (*str*) – Input PDB file

- **smile** (*str*) – Ligand's SMILE
- **ff** (*str, default='amber99sb-ildn'*) – Forcefield

run(*lambda_coul_list*, *lambda_vdw_list*, *lambda_restr_list=[]*, *mbar=False*, *dir_name='free_ener_run'*, *em_steps=5000*, *nvt_time=10*, *npt_time=10*, *prod_time=100*, *dt=0.002*, *temp=300.0*, *temp_groups='Protein non-Protein'*, *maxwarn=1*, *monitor_tool={'file_check_ext': 'log', 'function': <function progress_bar>}*)

Compute free energy to uncouple a molecule to a system.

Parameters

- **dir_name** (*str, default='free_ener_run'*) – path of the output folder
- **mol_name** (*str*) – Name of the molecule
- **lambda_coul_list** (*list*) – List lambda points for Coulomb
- **lambda_vdw_list** (*list*) – List lambda points for Lennard Jones
- **lambda_bond_list** (*list, default=[]*) – List lambda points for restraints
- **mbar** (*bool, default=False*) – MBAR flag
- **em_steps** (*int, default=5000*) – number of minimisation steps
- **nvt_time** (*int, default=10 ps*) – Time (ps) of NVT equilibration
- **npt_time** (*int, default=10 ps*) – Time (ps) of NPT equilibration
- **prod_time** (*float, default=100 ps*) – Time (ps) of production run
- **dt** (*float, default=0.002*) – integration time step
- **name** (*str, default=None*) – name of the simulation to run
- **temp** (*float, default=300.0*) – Temperature K
- **temp_groups** (*str, default='Protein non-Protein'*) – Group(s) for temperature coupling
- **maxwarn** (*int, default=0*) – Maximum number of warnings when using `gmx grompp`
- **monitor** – option to monitor a simulation, if not none monitor should contains two values: `function` the function to be ran while simulation is running and `input` parameters for the function

Object requirement(s):

- self.gmxsys.coor_file
- self.gmxsys.top_file
- self.gmxsys.nt
- self.gmxsys.ntmpi
- self.gmxsys.gpu_id

Object field(s) changed:

- self.lambda_sys_list
- self.lambda_coul
- self.lambda_vdw
- self.lambda_restr

- self.prod_time
- self.xvg_file_list

show_intermol_restr()

Show traj with atom implied in intermolecular restraints using nglview library.

static symmetry_correction(*smile*, *temp=300*)

Compute symmetry correction $\Delta_{sym} = kTln(\sigma)$

return value in kcal/mol

```
> FreeEner.symmetry_correction('c1ccccc1')
-1.4814...
```

property unit_graph

Return unit as math latex for matplotlib label purpose.

property unit_name

water_box_from_SMILE(*smile*, *method_3d='rdkit'*, *iter_num=5000*, *box_dist=1.1*)

Create a water box coordinates and toplogie with a molecule defined by its SMILE.

Parameters

- **smile** (*str*) – Molecule's SMILE
- **method_3d** (*str, default='rdkit'*) – Method to compute 3D coordinates
- **iter_num** (*int, default=5000*) – Iteration step number for 3D coordinate computation
- **box_dist** (*float, default=1.1 nm*) – Ditance to egde box (nm)

Note: Default box dist 1.1 nm was taken as minimal distance for CH4 molecule, to allow domain decomposition with *gmx mdrun*.

gromacs_py.free_ener.**show_log**()

To use only with Doctest !!! Redirect logger output to sys.stdout

13.1.4 Module contents

CHAPTER

FOURTEEN

GROMACS_PY

Gromacs_py is a Python library allowing a simplified use of the Gromacs MD simulation software. **Gromacs_py** can build a system topologie based on a pdb file, create the simulation system (pbc box, adding water and ions) and run minimisation, equilibration and production runs. One of the main objective of the **Gromacs_py** wrapper is to automatize routine operations for MD simulation of multiple systems.

Gromacs_py is under active development using continuous integration with Travis Cl.

- **Online Documentation:**
  https://gromacs-py.readthedocs.io
- **Source code repository:**
  https://github.com/samuelmurail/gromacs_py

14.1 Quick install

The latest release can be installed via *pip* or *conda*.

14.1.1 Conda

If you don't need a GPU compiled version of Gromacs you can use directly the **Gromacs_py** conda package to install both Gromacs software and **Gromacs_py** library:

```
conda install -c bioconda gromacs_py
```

14.1.2 Pip (Deprecated)

If the folowing softwares/modules are installed then you need to install the **Gromacs_py** library using pypi :

```
pip install gromacs_py
```

- Gromacs (version >= 5.1)
- Ambertools
- Rdkit
- Acpype

and add the software path Gromacs in the environmnent variable `$PATH`, *eg.* for gromacs:

```
# Add gromacs 'gmx' path:
export PATH='*path_to_gromacs*/bin/':$PATH
```

14.2 Authors

- Samuel Murail, Associate Professor - Université Paris Diderot, CMPLI.

See also the list of contributors who participated in this project.

14.3 License

This project is licensed under the GNU General Public License v2.0 - see the `LICENSE` file for details.

14.4 Indices and tables

- genindex
- modindex
- search

PYTHON MODULE INDEX

g

INDEX

A

C

I

L

M

N

O

P

R

S

T

U

V

W

X