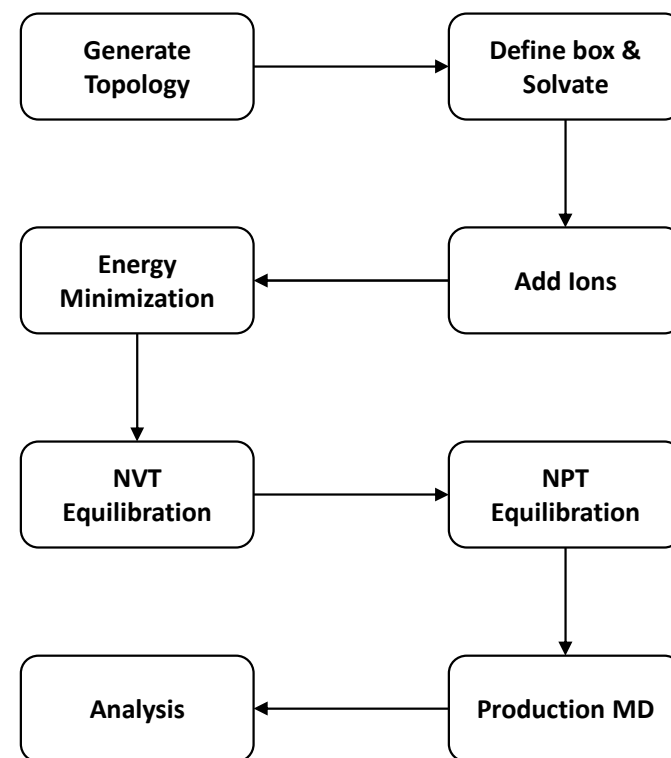
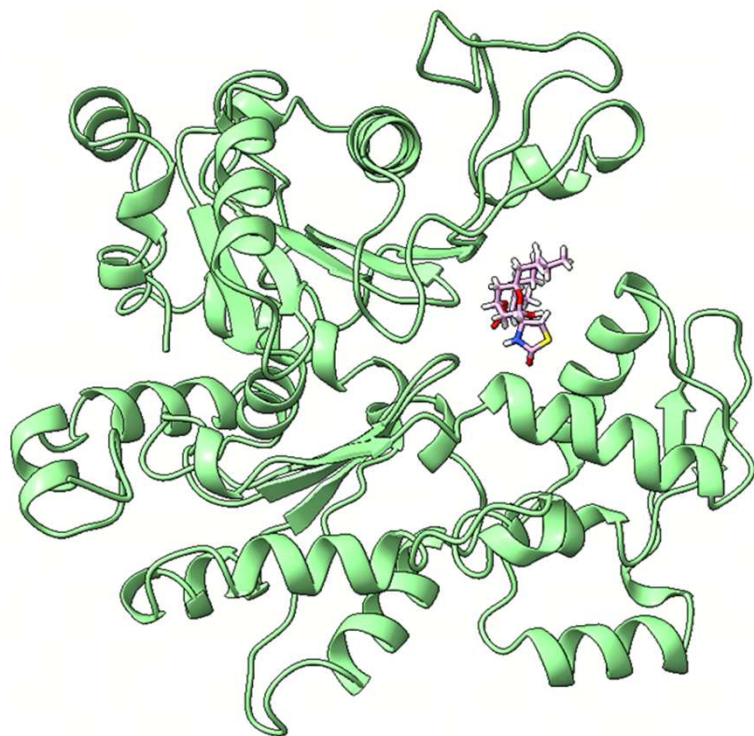


# **Simulation of Protein-Ligand Complex**

Naf Guo  
2025

# Protein-Ligand Complex Simulation

---



# Download PDB file

这里选用4b1y来进行示范

RCSB PDB Deposit Search Visualize Analyze Download Learn About Careers COVID-19 Help Contact us MyPDB

RCSB PDB PROTEIN DATA BANK 232,059 Structures from the PDB 1,068,577 Computed Structure Models (CSM)

Enter search term(s), Entry ID(s), Ligand ID or sequence Include CSM ?

Advanced Search | Browse Annotations Help

PDB-101 PDB EMDDataResource NAKB wwPDB Foundation PDB-IHM

Structure Summary Structure Annotations Experiment Sequence Genome Ligands Versions

Biological Assembly 1

4B1Y

Structure of the Phactr1 RPEL-3 bound to G-actin

PDB DOI: <https://doi.org/10.2210/pdb4B1Y/pdb>

Classification: **STRUCTURAL PROTEIN**

Organism(s): *Oryctolagus cuniculus*, *Mus musculus*

Expression System: *Escherichia coli* BL21

Mutation(s): No

Deposited: 2012-07-12 Released: 2013-07-31

Deposition Author(s): Moulleron, S., Wleziak, M., O'Reilly, N., Treisman, R., McDonald, N.Q.

Experimental Data Snapshot

Method: X-RAY DIFFRACTION

Resolution: 1.29 Å

R-Value Free: 0.174 (Depositor), 0.180 (DCC)

R-Value Work: 0.150 (Depositor), 0.150 (DCC)

R-Value Observed: 0.151 (Depositor)

Starting Model: experimental

View more details

wwPDB Validation

Metric	Value
Rfree	0.175
Clashscore	7
Ramachandran outliers	0
Sidechain outliers	0.6%
RSRZ outliers	5.1%

Display Files Download Files Data API

- FASTA Sequence
- PDBx/mmCIF Format
- PDBx/mmCIF Format (gz)
- BinaryCIF Format (gz)
- PDB Format**
- PDB Format (gz)
- PDBML/XML Format (gz)
- Structure Factors (CIF)
- Structure Factors (CIF - gz)
- Validation Full PDF
- Validation (XML - gz)
- Validation (CIF - gz)
- Validation 2fo-fc coefficients (CIF - gz)
- Validation fo-fc coefficients (CIF - gz)
- Biological Assembly 1 (CIF - gz)
- Biological Assembly 1 (PDB - gz)

Full Report

Global Symmetry: Asymmetric - C1

Global Stoichiometry: Hetero 2-mer - A1B1

登录PDB结构数据库

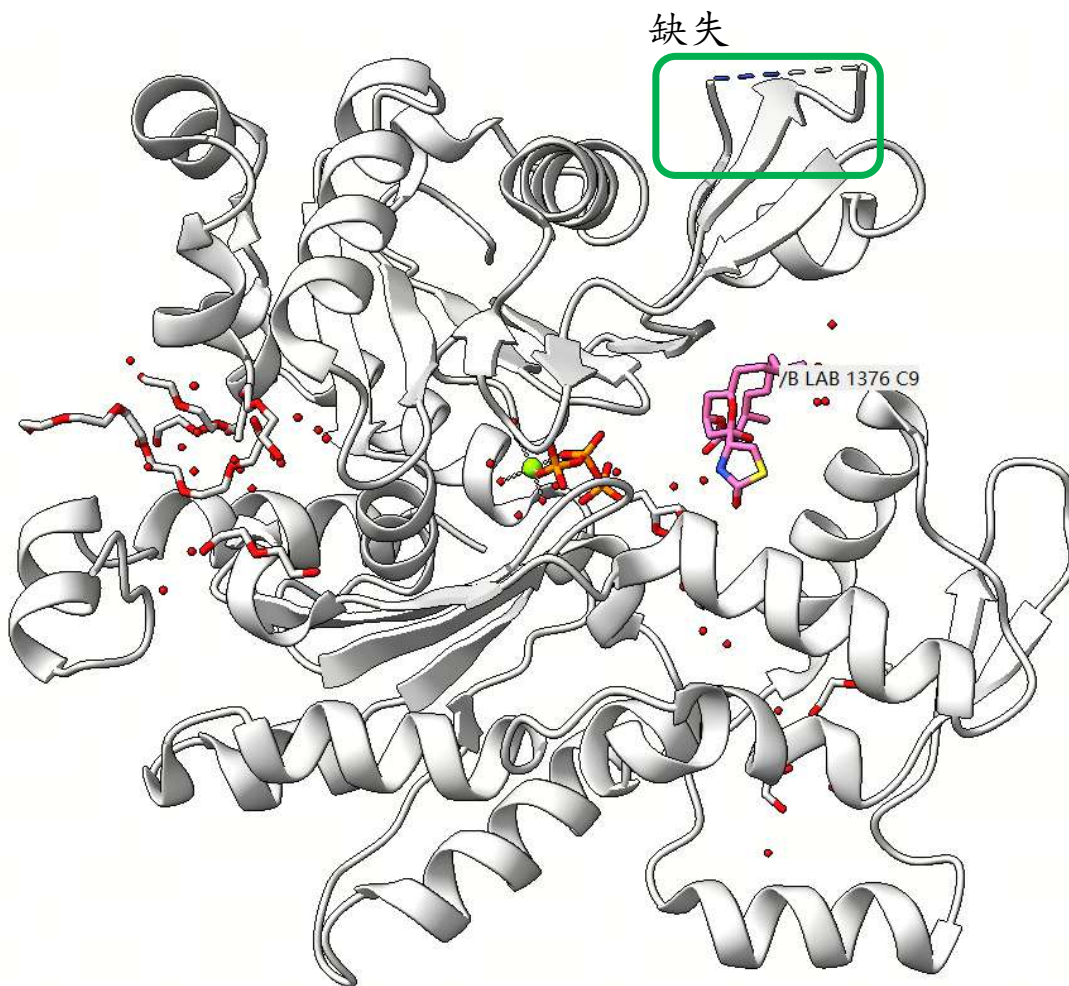
<https://www.rcsb.org/>

搜索PDB号 4B1Y

下载PDB格式文件

## See the PDB file

<https://www.cgl.ucsf.edu/chimerax/>



使用软件如ChimeraX, pymol检查4b1y, 我们会发现这个结构有一小段序列缺失了。结构内除了一个**大环配体**外, 还有一些脂质, 丙三醇, ATP, 水以及一个金属离子。

一般情况下, 脂质和丙三醇与蛋白质的功能无关, 我们可以删除;

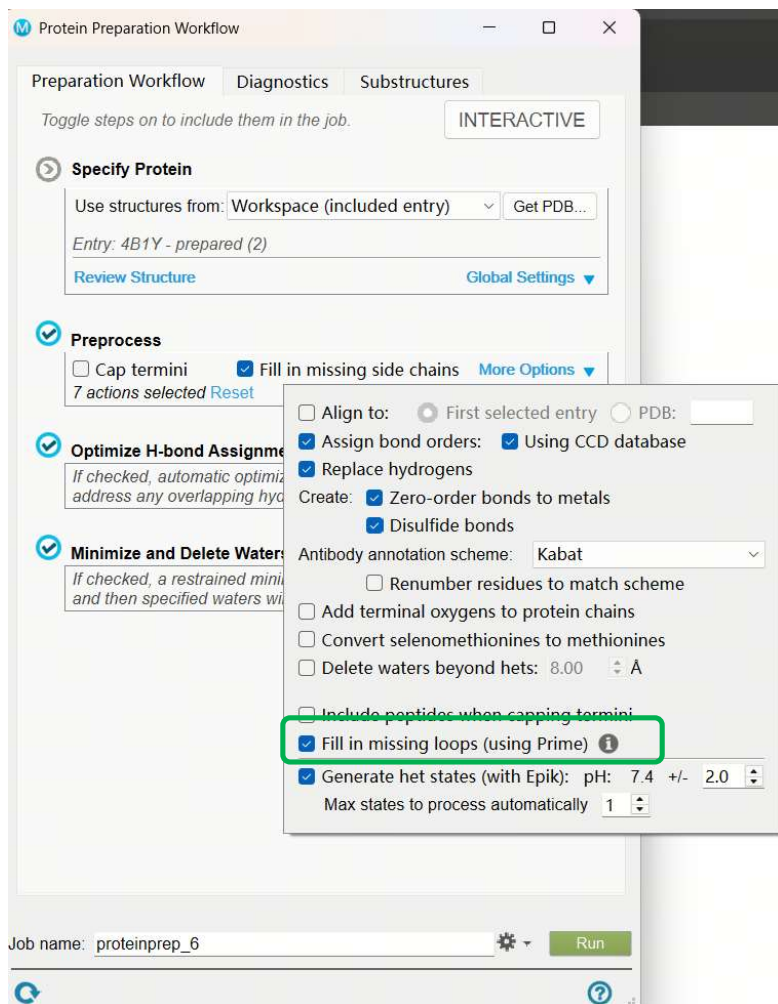
ATP和金属离子通常是重要的, 但这里为了简单起见, 也删除掉;

一般模拟的时候我们会保留配体周围6埃以内的结晶水, 这里简单起见, 也先删了。

这样我们只留蛋白和一个配体分子(注意把鼠标放到配体上会发现它叫LAB, 别删错了)。

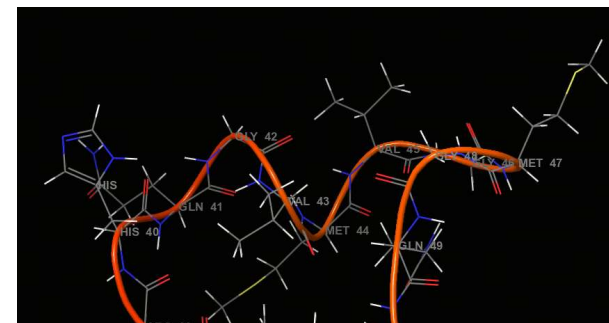
# Protein Preparation (with Maestro)

如果你的学校或者实验室购买了Schrodinger软件的话，直接使用Protein Preparation Workflow，可以比较顺利地完  
成蛋白质和小分子配体的准备工作。



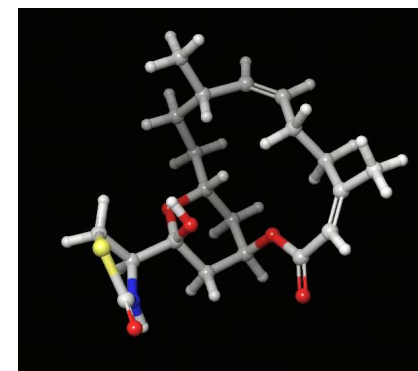
蛋白方面

缺失的序列被填充；  
缺失的氨基酸侧链被补齐；  
缺失的氢也都加上了



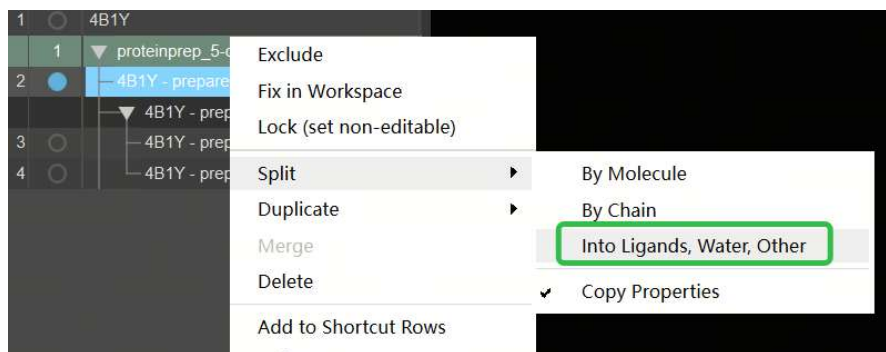
配体方面

键级被正确识别；  
预测得到pH=7.4的质子化状态；  
缺失的氢也都加上了

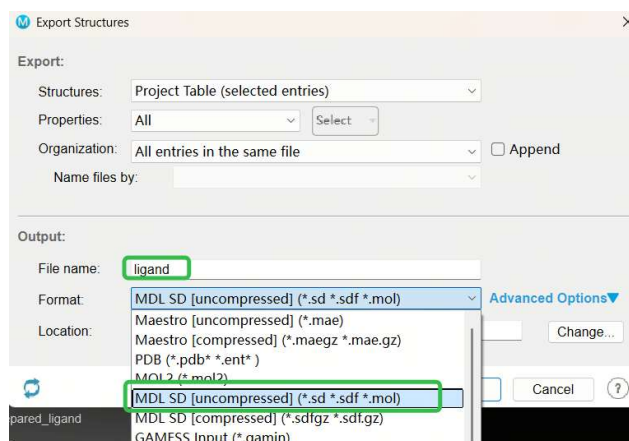
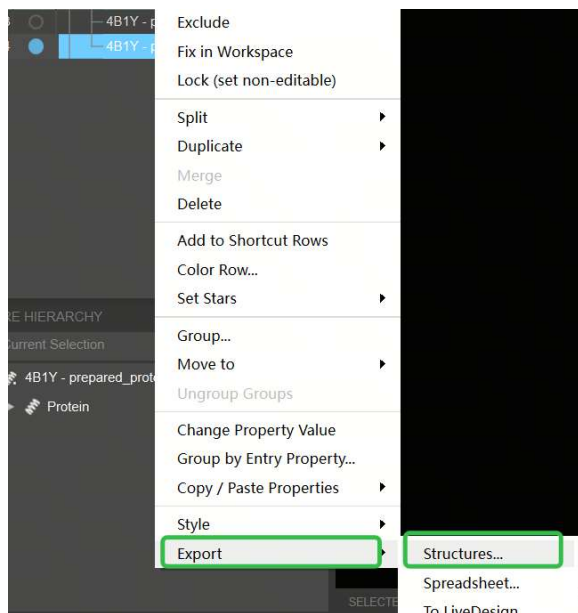




# Protein Preparation (with Maestro)



之后右键点击准备完的蛋白质entry，按Ligand，Water，Other可以把蛋白质和配体拆开



然后将蛋白导出为PDB格式，将配体导出为sdf格式，我们就做好了体系的准备工作了

## Protein Preparation (without Maestro)

但是，你可能没有Schrodinger可以用，我们得想办法找替代，首先是蛋白，我们可以使用pdbfixer

<https://htmlpreview.github.io/?https://github.com/openmm/pdbfixer/blob/master/Manual.html>

pdbfixer是一个python包，我们把它安装在之前安装gromacs时建立的环境里

```
conda activate gcc
```

```
conda install -c conda-forge pdbfixer openmm
```

然后在命令行敲pdbfixer，用网页浏览器打开 <http://localhost:8000/> 就能看到pdbfixer的界面了

### Welcome To PDBFixer!

Select a PDB file to load. It will be analyzed for problems.

☒ Load a local file

PDB File:  未选择任何文件

☐ Download a file from RCSB

PDB Identifier:

Analyze File

Chain	Residue Positions	Sequence	Add?
B	0 to 0	CYS	<input type="checkbox"/>
B	41 to 51	GLN, GLY, VAL, MET, VAL, GLY, MET, GLY, GLN, LYS, ASP	<input checked="" type="checkbox"/>
M	523 to 524	SER, ASP	<input checked="" type="checkbox"/>

Chain	Residue	Missing Atoms
B	HIS 40	CG, ND1, CE1, NE2, CD2
B	LYS 113	CG, CD, CE, NZ
B	GLU 270	CG, CD, OE1, OE2
B	GLU 334	CG, CD, OE1, OE2
B	GLU 364	CG, CD, OE1, OE2
M	LYS 493	CG, CD, CE, NZ
M	GLU 515	CG, CD, OE1, OE2

把下载的4b1y结构上传，我们只要蛋白，顺着流程往下自然走，会发现这个结构缺失了一系列残基和侧链，总之很自然地修好了这些问题，存出protein.pdb

如果你的pdbfixer装在了远程环境，没办法看图形界面，那么用下面类似的代码也可以完成蛋白质修复

```
pdbfixer 4b1y.pdb --output protein.pdb --keep-heterogens=none --add-residues --verbose
```

# Protein Preparation (without Maestro)

还有一种办法，就是用Swiss-Model建模

通过PDB数据库里找到蛋白的UniProt ID，链接到UniProt数据库

Macromolecules

Find similar proteins by: Sequence (by identity cutoff) | 3D Structure

Entity ID: 1

Molecule	Chains	Sequence Length	Organism	Details	Image
ACTIN, ALPHA SKELETAL MUSCLE	A [auth B]	376	<a href="#">Oryctolagus cuniculus</a>	Mutation(s): 0 EC: <a href="#">3.6.4</a>	

UniProt

Find proteins for [P68135](#) (*Oryctolagus cuniculus*)

Explore [P68135](#)

Go to UniProtKB: [P68135](#)

← → ↻ uniprot.org/uniprot/[P68135](#)/entry#sequences

Bing 计算化学公社 - 高... Computational Ch... McCommon Group Brian Shoichet, Ph... Atomic S

UniProt BLAST Align Peptide search ID mapping SPARQL UniProtKB

Function

Names & Taxonomy

Subcellular Location

Phenotypes & Variants

PTM/Processing

Expression

Interaction

Structure

Family & Domains

**Sequence**

Similar Proteins

Entry Variant viewer Feature viewer Genomic coordinates

See also sequence in UniParc or sequence clusters in UniRef

Tools [Download](#) [Add](#) [Highlight](#) [Copy sequence](#)

Length 377  
Mass (Da) 42,051

10 20 30 40  
MCDEDETTAL VCDNGSGLVK AGFAGDDAPR AVFPSIVGRP RHQGMVM

120 130 140 150  
PLNPKANREK MTQIMFETFN VPAMYVAIQ VLSLYASGRT TGIVLDS

230 240 250 260  
VALDFENEMA TAASSSSLEK SYELPDGQVI TIGNERFRCP ETLFQPSI

340 350 360 370  
IIAPPERKYS VWIGGSILAS LSTFQQMWIT KQEYDEAGPS IVHRKCF

在UniProt数据库里可以获得蛋白质的序列

```
>sp|P68135|ACTS_RABIT Actin, alpha skeletal muscle OS=Oryctolagus cuniculus OX=9986 GN=ACTA1 PE=1 SV=1
MCDEDETTALVCDNGSGLVKAGFAGDDAPRAVFPISVGRPRHQGMVMGMGQKDSYVGDEA
QSKRGILTLKYPIDHGIITNWDDMEKIWHHTFYNELRVAPEEHPTLLTEAPLNPKANREK
MTQIMFETFNVPAMYVAIQAVLSLYASGRTTGIVLDSGDGVTHNVPYIEGYALPHAIMRL
DLAGRDLDYLMKILTERGYSFVTTAEREIVRDIKEKLCYVALDFENEMATAASSSSLEK
SYELPDGQVITIGNERFRCPETLFQPSFIGMESAGIHETTYNSIMKCDIDIRKDLANNV
MSGGTTMYPGIADRMQKEITALAPSTMKIKIAPPKYSVWIGGSILASLSTFQQMWIT
KQEYDEAGPSIVHRKCF
```



# Protein Preparation (without Maestro)

<https://swissmodel.expasy.org/interactive>

打开Swiss-Model，切换到User Template模式，输入蛋白质的序列，提交我们下载的结构作为Template，等于是用这个蛋白自己的结构给它自己建模，这样补齐缺失的部分

Start a New Modelling Project

Target Sequence(s):  
(Format must be FASTA, Clustal, plain string, or a valid UniProtKB AC)

Target

MCDEETALVCDNGSGLVKAGFAGDDAPRAVTPSIVGRPRHQGVNVGMGQKDSYVGDQAQSKRGILTLKYPIEHGIITNWDDMEKIWHHTFYNELRVAPEEHPTLLTEAPLNPKANREKMTQIMFETFNVPAMYVAIQAVLSLYASGRT

150

Target

TGIVLDSGDGVTHNVPIYEGYALPHAIMRLDLAGRDLTDYLMKILTERGYSFVTTAEREIVRDIKEKLCYVALDFENEMATAASSSSLEKSYELPDGQVITIGNERFRCPETLFQPSFIGMESAGIHETTYSIMKCDIDIRKDYANNV

300

Target

MSGGTTMYPGIADRMQKEITALAPSTMKIKIIPPERKYSVWIGGSILASLSTFQQMWITKQEYDEAGPSIVHRKCF

377

Add Hetero Target

Reset

Supported Inputs

Sequence(s)

Target-Template Alignment

User Template

DeepView Project

Template File:

+ Add Template File...

Project Title:

Untitled Project

Email:

Optional

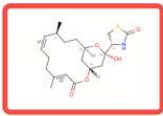

Build Model

By using the SWISS-MODEL server, you agree to comply with the following [terms of use](#) and to cite the corresponding [articles](#).

任务跑完以后把得到的model下载下来，删掉蛋白质之外的部分，这样蛋白部分勉强也算是修好了

## Protein Preparation (without Maestro)

修完蛋白还需要修配体，配体的主要问题是没有氢，而且PDB里可能没带配体的质子化状态，化学键键级等信息  
先在PDB网站上把配体的sdf或者mol2格式的文件下载下来

<b>LAB</b> <a href="#">Query on LAB</a>  Download Ideal Coordinates CCD File ④ Download Instance Coordinates ▾ <b>SDF format, chain C [auth B]</b> <b>MOL2 format, chain C [auth B]</b> mmCIF format, chain C [auth B]	C [auth B]	<b>LATRUNCULIN B</b> C <sub>20</sub> H <sub>29</sub> N O <sub>5</sub> S NSHPHXHGRHSMIK-JRIKCGFMSA-N		Interactions ▾ Interactions & Density ▾
	K [auth M]	<b>HEXAETHYLENE GLYCOL</b> C <sub>12</sub> H <sub>26</sub> O <sub>7</sub> IIRDTKBZINWQAW-UHFFFAOYSA-N		Interactions ▾ Interactions & Density ▾

激活环境，安装需要的python包

```
conda activate gcc
```

```
conda install conda-forge::openbabel
```

```
pip install Xponge rdkit
```

使用obabel按pH=7.4 的情况给下载下来的配体mol2文件加氢，处理键级

```
obabel 4b1y_C_LAB.mol2 -O LAB.mol2 -p 7.4
```

按照PDB文件里配体分子的名字修改mol2文件里原子的名字

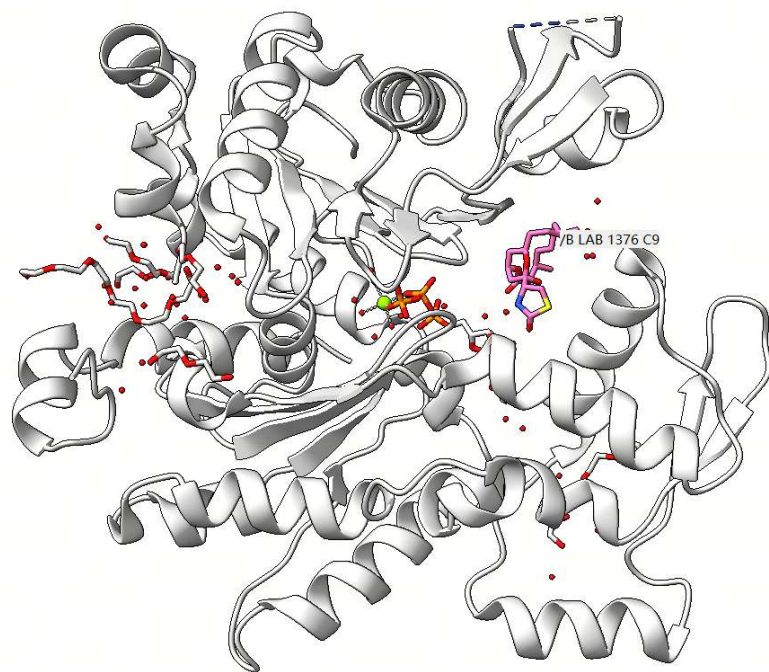
```
Xponge name2name -tformat pdb -tfile 4b1y.pdb -tres LAB -fformat mol2 -ffile LAB.mol2 -oformat mol2 -ofile LAB.mol2
```

顺利的话得到的LAB.mol2就是我们准备好的配体文件了

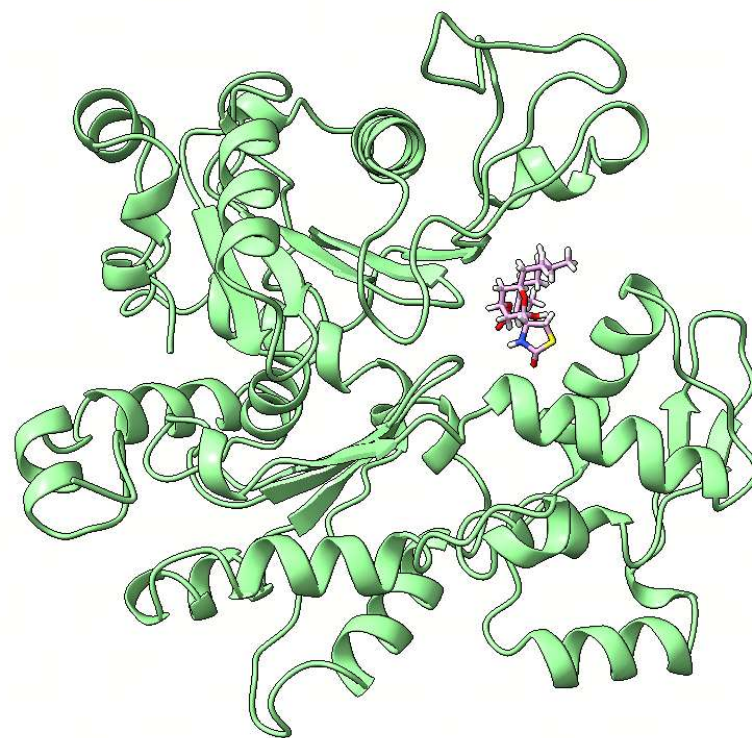
## Protein Preparation (without Maestro)

把pdbfixer修复得到的protein.pdb和obabel处理得到的LAB.mol2同时用ChimeraX打开，效果意外的挺好的

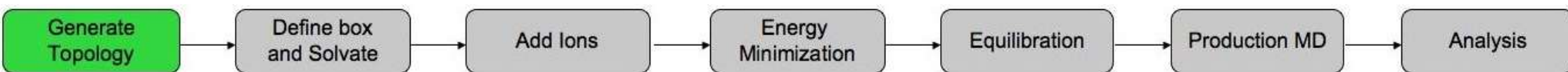
处理前



处理后



## Generate Topology



在上面，我们修复了蛋白质的各种问题，但其实还有一个步骤，就是处理蛋白质的质子化状态

再使用maestro准备蛋白质时，虽然maestro使用proPKa预测了残基带电情况，添加了氢原子，但是gromacs可能不认识，我们需要根据maestro处理后的文件修改PDB里对应残基的名字。总之我提供了一个脚本来做处理。

<https://github.com/Sept-naf/gromacs-tutorials/blob/main/maestro2amber.py>

```
python maestro2amber.py maestro_format.pdb amber_readable.pdb
```

有没有maestro其实也可以用另一种方法处理，pdb2pqr

<https://pdb2pqr.readthedocs.io/en/latest/index.html>

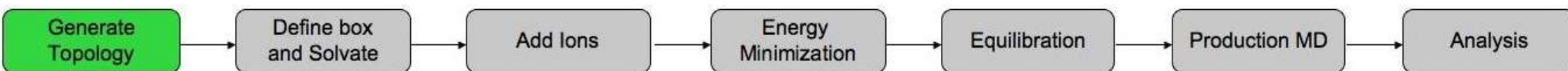
```
pip install pdb2pqr -i https://pypi.tuna.tsinghua.edu.cn/simple
```

对一个已经使用maestro或者pdbfixer处理好了的pdb文件

```
pdb2pqr input.pdb output.pdb --ffout AMBER --with-ph 7.4
```

此时得到的文件就带有gromacs可以识别的pH7.4下的质子化状态

## Generate Topology



可是，上一页我们到底做了什么呢？

ATOM	76	N	HIE	6	32.889	22.953	0.735	-0.4000	1.5000
ATOM	77	CA	HIE	6	31.659	23.744	0.748	-0.0000	2.0000
ATOM	78	C	HIE	6	30.798	23.343	1.936	0.5500	1.7000
ATOM	79	O	HIE	6	31.194	22.404	2.625	-0.5500	1.4000
ATOM	80	CB	HIE	6	30.933	23.526	-0.586	0.1250	2.0000
ATOM	81	CG	HIE	6	31.716	24.056	-1.802	0.1550	1.7000

ATOM	439	N	HID	29	26.233	24.528	19.268	-0.4000	1.5000
ATOM	440	CA	HID	29	24.981	24.221	19.961	-0.0000	2.0000
ATOM	441	C	HID	29	23.882	23.906	18.957	0.5500	1.7000
ATOM	442	O	HID	29	23.903	24.446	17.842	-0.5500	1.4000
ATOM	443	CB	HID	29	24.635	25.417	20.826	0.1250	2.0000
ATOM	444	CG	HID	29	25.736	25.621	21.859	-0.1250	1.7000

ATOM	34	N	CYX	3	40.248	17.802	0.586	-0.4000	1.5000
ATOM	35	CA	CYX	3	39.209	18.135	1.542	-0.0000	2.0000
ATOM	36	C	CYX	3	37.962	18.602	0.826	0.5500	1.7000
ATOM	37	O	CYX	3	38.007	19.100	-0.321	-0.5500	1.4000
ATOM	38	CB	CYX	3	39.687	19.276	2.433	0.2900	2.0000
ATOM	39	SG	CYX	3	41.285	18.930	3.230	-0.2900	1.8500
ATOM	40	H	CYX	3	40.531	18.460	-0.188	0.4000	1.0000
ATOM	41	HA	CYX	3	38.981	17.305	2.064	0.0000	0.0000
ATOM	42	HB2	CYX	3	39.792	20.089	1.876	0.0000	0.0000
ATOM	43	HB3	CYX	3	39.014	19.422	3.147	0.0000	0.0000

其实就是根据质子化状态改变残基名字，比如，根据实际质子化情况把HIS改名为HIE，HID，HIP；

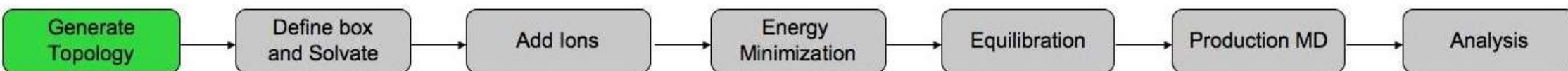
同类的有GLU，GLH；ASP，ASH等。

还有一个就是形成了二硫键的CYS会被改名为CYX

所以，其实你是可以手动去编辑PDB文件来完成质子化状态的编辑的。



## Generate Topology



在之前的步骤中，我们分别准备好了蛋白质protein.pdb和配体，如果你用的maestro，配体应该存了sdf；用obabel应该存了mol2格式  
现在，我们要分别对蛋白质和配体产生拓扑和坐标文件

首先是蛋白

```
gmx pdb2gmx -f protein.pdb -o protein.gro -p topol.top -ignh
```

检查生成的结果，会发现此命令产生了protein.gro（蛋白质坐标文件）；topol.top（拓扑文件）；由于我们选的蛋白有两条链B和M，gromacs分别产生了B链和M链的力场文件和限制文件

```
(gcc) root@bohrium-11312-1251688:~/complex_md# ls
4b1y.pdb          LAB.mol2  posre_Protein_chain_B.itp  protein.gro  test.pdb  topol_Protein_chain_B.itp
4b1y_C_LAB.mol2  fixed.pdb posre_Protein_chain_M.itp  protein.pdb  topol.top  topol_Protein_chain_M.itp
```

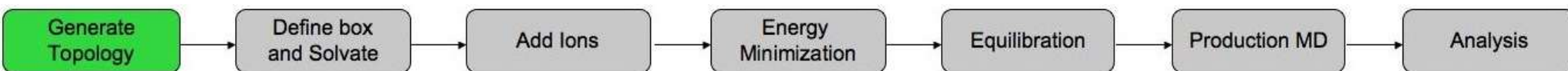
posre\_Protein\_chain\_B.itp和posre\_Protein\_chain\_M.itp很好理解，就是对每个原子施加限制的文件，在后续体系平衡的时候会被使用；那么这里的topol\_Protein\_chain\_B.itp和topol\_Protein\_chain\_M.itp是什么？和topol.top有什么关系呢？

```

In this topology include file, you will find position restraint
; entries for all the heavy atoms in your original pdb file.
; This means that all the protons which were added by pdb2gmx are
; not restrained.

[ position_restraints ]
; atom  type      fx      fy      fz
   1      1    1000    1000    1000
   5      1    1000    1000    1000
   7      1    1000    1000    1000
  10      1    1000    1000    1000
  12      1    1000    1000    1000
  13      1    1000    1000    1000
  14      1    1000    1000    1000
  16      1    1000    1000    1000
```

# Generate Topology



```
; Include forcefield parameters
#include "amber99sb-ildn.ff/forcefield.itp"

; Include chain topologies
#include "topol_Protein_chain_B.itp"
#include "topol_Protein_chain_M.itp"

; Include water topology
#include "amber99sb-ildn.ff/tip3p.itp"

#ifdef POSRES_WATER
; Position restraint for each water oxygen
[ position_restraints ]
; i funct      fcx      fcy      fcz
  1    1      1000      1000      1000
#endif

; Include topology for ions
#include "amber99sb-ildn.ff/ions.itp"

[ system ]
; Name
Protein

[ molecules ]
; Compound      #mols
Protein_chain_B      1
Protein_chain_M      1
```

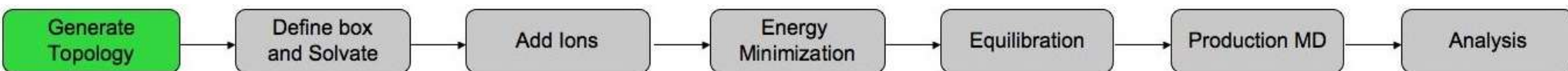
打开topol.top, 检查其内容, 首先引用了forcefield.itp, 我们在蛋白模拟中已经见过

然后注意到这里引用了topol\_Protein\_chain\_B.itp和topol\_Protein\_chain\_M.itp, 其实已经可以猜测, 就和tip3p.itp记录了水的力场参数和原子连接方式一样, topol\_Protein\_chain\_B.itp和topol\_Protein\_chain\_M.itp应该分别记录了链B和链M的力场参数和原子之间的连接方式

水的力场文件, 水的限制, 离子的力场文件, 我们也都见过了

[ molecules ]字段, 记录了此时的体系也就是protein.gro里有2个物种, 分别叫Protein\_chain\_B和Protein\_chain\_M, 数目均为1个, 且在gro文件中出现的先后顺序是B链先, M后

# Generate Topology



```
[ moleculetype ]
; Name          nrexcl
Protein_chain_B 3

[ atoms ]
; nr      type  resnr residue  atom  cgnr      charge      mass  typeB      chargeB      massB
; residue  0 CYS rtp NCYS q +1.0
; 1        N3    0    CYS      N      1      0.1325      14.01
; 2        H     0    CYS      H1     2      0.2023      1.008
; 3        H     0    CYS      H2     3      0.2023      1.008
```

```
[ bonds ]
; ai  aj  funct
; 1  2    1
; 1  3    1
; 1  4    1
; 1  5    1
; 5  6    1
```

```
[ angles ]
; ai  aj  ak  funct
; 2  1  3    1
; 2  1  4    1
; 2  1  5    1
; 3  1  4    1
; 3  1  5    1
```

```
[ dihedrals ]
; ai  aj  ak  al  funct
; 2  1  5  6    9
; 2  1  5  7    9
; 2  1  5  12   9
; 3  1  5  6    9
; 3  1  5  7    9
; 3  1  5  12   9
```

```
; Include Position restraint file
#ifdef POSRES
#include "posre_Protein_chain_B.itp"
#endif
```

打开topol\_Protein\_chain\_B.itp，检查其内容

首先看到了[moleculetype]字段，记录了这个物种的名字Protein\_chain\_B，和topol.top里[molecules]字段的物种名对应；之后出现了[atoms]字段，记录了链B所有原子的PDB名，残基序号，残基名，在力场里的原子类型，以及电和质量等信息

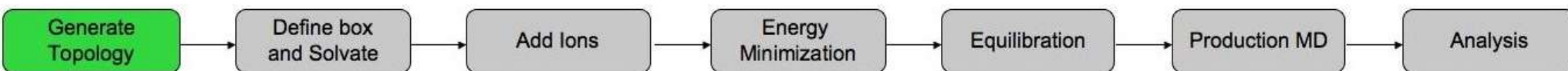
继续往下看，会看到熟悉的[bonds]字段，记录了所有的键连方式；以及[angles]字段，记录键角；二面角字段[dihedrals]

翻到最后会看到引用了链B的限制文件

posre\_Protein\_chain\_B.itp，在上两期纯蛋白的模拟中我们知道，当mdp文件中出现-DPOSRES时，这个限制文件就会被启用

所以，如我们所预期的，topol\_Protein\_chain\_B.itp和topol\_Protein\_chain\_M.itp就是链B和链M的力场文件

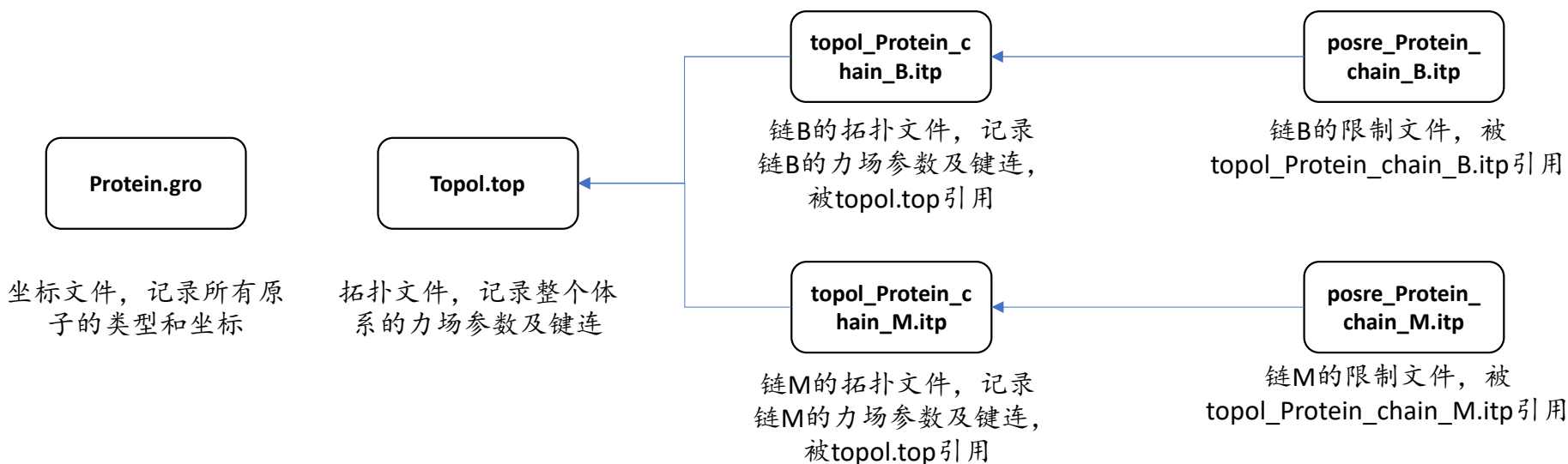
## Generate Topology



```
gmx pdb2gmx -f protein.pdb -o protein.gro -p topol.top -ignh
```

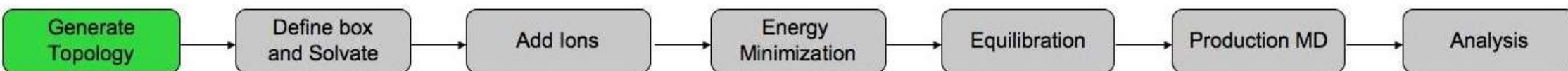
```
(gcc) root@bohrium-11312-1251688:~/complex_md# ls
4b1y.pdb      LAB.mol2  posre_Protein_chain_B.itp  protein.gro  test.pdb  topol_Protein_chain_B.itp
4b1y_C_LAB.mol2  fixed.pdb  posre_Protein_chain_M.itp  protein.pdb  topol.top  topol_Protein_chain_M.itp
```

小结一下，我们使用pdb2gmx命令处理具有B和M两条链的protein.pdb，得到了6个文件，它们之间的关系是





## Generate Topology



蛋白处理好了，该准备配体的力场参数了，在之前的蛋白质和配体准备步骤中，我们用maestro处理得到了LAB.sdf；或者用obabel处理得到了LAB.mol2，在后续的处理中，用哪个都一样

因为蛋白使用了amber力场，所以配体要用对应的gaff2力场参数，我们需要安装对应的软件acpype

```
conda activate gcc
```

首先激活用于MD的虚拟环境

```
conda install -c conda-forge acpype
```

安装acpype及其依赖

然后就可以处理小分子了

```
acpype -i LAB.mol2 -c bcc -a gaff2 -n 0 -b MOL
```

-i LAB.mol2 用LAB.mol2作为输入文件

-c bcc 用bcc方法计算每个原子的电荷（这个方法精度一般，能用；现在很多人会计算RESP电荷）

-a gaff2 小分子力场参数从与amber力场匹配的gaff2数据集获取

-n 0 小分子带电为0，这里一定要根据我们的小分子的实际带电情况输入

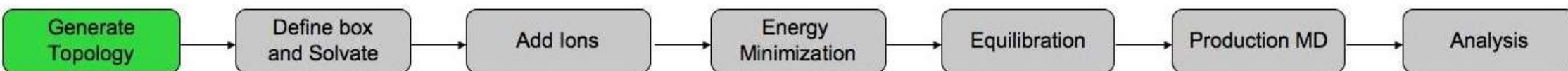
-b MOL 输出文件用MOL作为前缀

```
(gcc) root@bohrium-11312-1251688:~/complex_md# ls -lh
total 8.2M
-rw-r--r-- 1 root root 693K Feb 27 02:46 4b1y.pdb
-rw-r--r-- 1 root root 2.2K Feb 27 03:29 4b1y_C_LAB.mol2
-rw-r--r-- 1 root root 5.7K Feb 27 03:34 LAB.mol2
drwxr-xr-x 2 root root 4.0K Feb 28 00:03 MOL.acpype
-rw-r--r-- 1 root root 653K Feb 27 02:47 fixed.pdb
```

几分钟后，计算结束，我们会看到一个MOL.acpype的文件夹



# Generate Topology



打开MOL.acpype, 我们要用到3个文件: MOL\_GMX.gro, MOL\_GMX.itp, posre\_MOL.itp

```
(gcc) root@bohrium-11312-1251688:~/complex_md/MOL.acpype# ls
ANTECHAMBER_AC.AC          ANTECHAMBER_BOND_TYPE.AC0  MOL.pkl          MOL_CHARMM.inp  MOL_CNS.top      MOL_GMX_OPLS.top  leap.log          sqm.out
ANTECHAMBER_AC.AC0        ANTECHAMBER_PREP.AC       MOL_AC.frcmod    MOL_CHARMM.prm  MOL_GMX.gro      MOL_NEW.pdb       md.mdp           sqm.pdb
ANTECHAMBER_AM1BCC.AC     ANTECHAMBER_PREP.AC0      MOL_AC.inpcrd    MOL_CHARMM.rtf  MOL_GMX.itp      MOL_bcc_gaff2.mol2 posre_MOL.itp
ANTECHAMBER_AM1BCC_PRE.AC  ATOMTYPE.INF              MOL_AC.lib       MOL_CNS.inp     MOL_GMX.top      acpype.log        rungmix.sh
ANTECHAMBER_BOND_TYPE.AC  LAB.mol2                  MOL_AC.prmtp     MOL_CNS.par     MOL_GMX_OPLS.itp em.mdp            sqm.in
```

很自然地, 我们能估计到MOL\_GMX.gro是坐标文件, MOL\_GMX.itp是小分子的力场文件, posre\_MOL.itp是小分子的限制文件

```
MOL_GMX.gro created by acpype (v: 2023.10.27)
56
1 LAB 05 1 1.109 0.192 2.769
1 LAB C18 2 1.052 0.074 2.736
1 LAB N1 3 0.974 0.100 2.609
1 LAB C16 4 0.843 0.015 2.612
1 LAB C17 5 0.856 -0.076 2.741
1 LAB S1 6 0.957 -0.006 2.850
1 LAB C15 7 0.716 0.097 2.603
1 LAB 04 8 0.691 0.148 2.729
1 LAB C14 9 0.726 0.212 2.505
1 LAB C13 10 0.604 0.259 2.455
1 LAB 02 11 0.537 0.342 2.556
1 LAB C12 12 0.508 0.154 2.419
```

```
MOL_GMX.itp created by acpype (v: 2023.10.27) on Fri Feb 28 00:01:55 2025

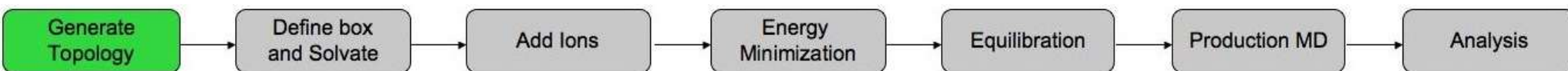
[ atomtypes ]
;name bond_type mass charge ptype sigma epsilon Amb
o o 0.00000 0.00000 A 3.04812e-01 6.12119e-01 ; 1.71 0.
c c 0.00000 0.00000 A 3.31521e-01 4.13379e-01 ; 1.86 0.
ns ns 0.00000 0.00000 A 3.26995e-01 4.91202e-01 ; 1.84 0.
c5 c5 0.00000 0.00000 A 3.39771e-01 4.51035e-01 ; 1.91 0.
ss ss 0.00000 0.00000 A 3.53241e-01 1.18156e+00 ; 1.98 0.
c6 c6 0.00000 0.00000 A 3.39771e-01 4.51035e-01 ; 1.91 0.
oh oh 0.00000 0.00000 A 3.24287e-01 3.89112e-01 ; 1.82 0.
os os 0.00000 0.00000 A 3.15610e-01 3.03758e-01 ; 1.77 0.
c3 c3 0.00000 0.00000 A 3.39771e-01 4.51035e-01 ; 1.91 0.
c2 c2 0.00000 0.00000 A 3.31521e-01 4.13379e-01 ; 1.86 0.
ce ce 0.00000 0.00000 A 3.31521e-01 4.13379e-01 ; 1.86 0.
hn hn 0.00000 0.00000 A 1.10650e-01 4.18400e-02 ; 0.62 0.
h1 h1 0.00000 0.00000 A 2.42200e-01 8.70272e-02 ; 1.36 0.
ho ho 0.00000 0.00000 A 5.37925e-02 1.96648e-02 ; 0.30 0.
hc hc 0.00000 0.00000 A 2.60018e-01 8.70272e-02 ; 1.46 0.
ha ha 0.00000 0.00000 A 2.62548e-01 6.73624e-02 ; 1.47 0.

[ moleculetype ]
;name nrexcl
MOL 3

[ atoms ]
; nr type resi res atom cgnr charge mass ; qtot bond_type
1 o 1 LAB 05 1 -0.573500 16.00000 ; qtot -0.573
2 c 1 LAB C18 2 0.715100 12.01000 ; qtot 0.142
3 ns 1 LAB N1 3 -0.555900 14.01000 ; qtot -0.414
4 c5 1 LAB C16 4 0.074700 12.01000 ; qtot -0.340
```

```
posre_MOL.itp created by acpype (v: 20
[ position_restraints ]
; atom type fx fy fz
1 1 1000 1000 1000
2 1 1000 1000 1000
3 1 1000 1000 1000
4 1 1000 1000 1000
5 1 1000 1000 1000
6 1 1000 1000 1000
```

# Generate Topology



`mv MOL_GMX.gro ../`

`mv MOL_GMX.itp ../`

`mv posre_MOL.itp ../`

把这三个文件移动到之前有蛋白力场的文件夹里 ../是当前目录的上级目录的意思，mv 是move

```

(gcc) root@bohrium-11312-1251688:~/complex_md/MOL.acpype# mv MOL_GMX.gro ../
(gcc) root@bohrium-11312-1251688:~/complex_md/MOL.acpype# mv MOL_GMX.itp ../
(gcc) root@bohrium-11312-1251688:~/complex_md/MOL.acpype# mv posre_MOL.itp ../
(gcc) root@bohrium-11312-1251688:~/complex_md/MOL.acpype# cd ../
(gcc) root@bohrium-11312-1251688:~/complex_md# ls
4b1y.pdb      LAB.mol2      MOL_GMX.gro   fixed.pdb     posre_Protein_chain_B.itp  protein.gro   test.pdb      topol_Protein_chain_B.itp
4b1y_C_LAB.mol2 MOL.acpype     MOL_GMX.itp   posre_MOL.itp posre_Protein_chain_M.itp  protein.pdb   topol.top     topol_Protein_chain_M.itp
  
```

protein.gro		MOL_GMX.gro				
文件	编辑	查看				
523SER	O	6430	3.023	-0.044	-0.535	
524ASP	N	6431	3.081	-0.067	-0.642	
524ASP	H	6432	2.986	-0.096	-0.631	
524ASP	CA	6433	3.147	-0.097	-0.770	
524ASP	HA	6434	3.175	-0.175	-0.713	
524ASP	CB	6435	3.113	-0.145	-0.915	
524ASP	HB1	6436	3.176	-0.221	-0.933	
524ASP	HB2	6437	3.135	-0.068	-0.974	
524ASP	CG	6438	2.993	-0.195	-0.980	
524ASP	OD1	6439	2.883	-0.145	-1.012	
524ASP	OD2	6440	3.024	-0.314	-0.999	
524ASP	C	6441	3.208	0.022	-0.829	
524ASP	OC1	6442	3.269	-0.071	-0.774	
524ASP	OC2	6443	3.317	0.075	-0.852	
1 LAB	O5	1	1.109	0.192	2.769	
1 LAB	C18	2	1.052	0.074	2.736	
1 LAB	N1	3	0.974	0.100	2.609	
1 LAB	C16	4	0.843	0.015	2.612	
1 LAB	C17	5	0.856	-0.076	2.741	
1 LAB	S1	6	0.957	-0.006	2.850	

protein.gro

文件

编辑

查看

6443 + 56 = 6499

Georgetown Riga Oslo Madrid Amsterd

6499

OCYS	N	1	-0.630	2.332	0.987
------	---	---	--------	-------	-------

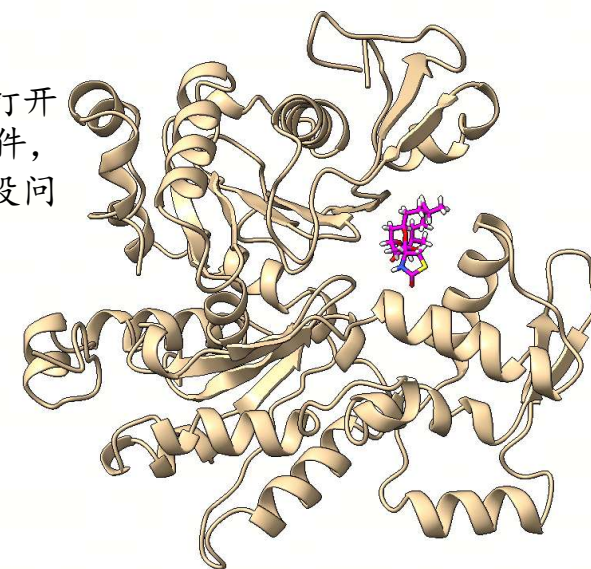
OCYS	H1	2	-0.630	2.296	0.894
------	----	---	--------	-------	-------

OCYS	H2	3	-0.572	2.414	0.991
------	----	---	--------	-------	-------

OCYS	H3	4	-0.594	2.263	1.049
------	----	---	--------	-------	-------

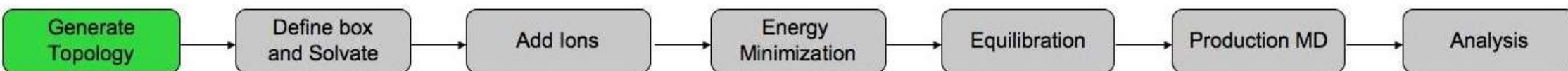
复制且仅复制MOL\_GMX.gro里原子的坐标列到protein.gro文件蛋白原子坐标列的后面；  
修改gro文件第二行的原子数为加上配体原子的数目

用可视化软件打开合并后的gro文件，  
确定相对位置没问题





# Generate Topology



处理完MOL\_GMX.gro以后处理力场和限制文件

```
; Include forcefield parameters
#include "amber99sb-ildn.ff/forcefield.itp"
#include "MOL_GMX.itp"

; Include Position restraint file
#ifdef POSRES
#include "posre_MOL.itp"
#endif

; Include chain topologies
#include "topol_Protein_chain_B.itp"
#include "topol_Protein_chain_M.itp"

; Include water topology
#include "amber99sb-ildn.ff/tip3p.itp"

#ifdef POSRES_WATER
; Position restraint for each water oxygen
[ position_restraints ]
; i funct      fcx      fcy      fcx
; 1 1          1000     1000     1000
#endif

; Include topology for ions
#include "amber99sb-ildn.ff/ions.itp"

[ system ]
; Name
Protein

[ molecules ]
; Compound      #mols
Protein_chain_B 1
Protein_chain_M 1
MOL              1
```

打开topol.top文件

- 首先，**必须紧跟着forcefield.itp的定义**，引入小分子的力场文件MOL\_GMX.itp;
- 其次，定义对小分子限制文件的引用;

```
[ moleculetype ]
;name      nrexcl
MOL        3
```

- 最后，在[molecules]字段里注册小分子物种和数目，注意这里的名字一定要与itp文件里[moleculetype]字段一致

## Define box & Solvate



我们终于准备好了力场，可以开始加水了

```
gmx editconf -f protein.gro -o protein_box.gro -c -d 1.5 -bt cubic
```

```
gmx solvate -cp protein_box.gro -cs spc216.gro -o solv.gro -p topol.top
```

```
; Include topology for ions
#include "amber99sb-ildn.ff/ions.itp"

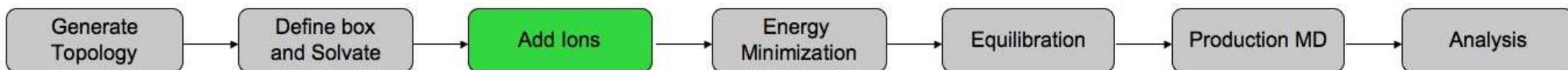
[ system ]
; Name
Protein in water

[ molecules ]
; Compound      #mols
Protein_chain_B  1
Protein_chain_M  1
MOL              1
SOL              32887
```

同样地，注意到topol.top里多了水

## Add Ions

---



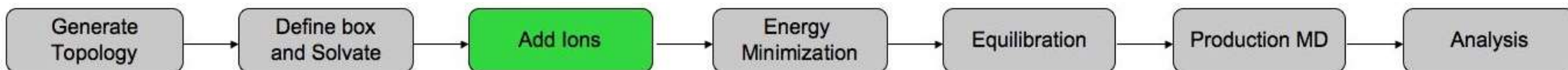
然后是加离子

```
gmx grompp -f ions.mdp -c solv.gro -p topol.top -o ions.tpr
```

```
gmx genion -s ions.tpr -o ions.gro -p topol.top -pname NA -nname CL -conc 0.15 -neutral
```



# Add Ions



mdp文件和之前没有区别

ions.mdp

```
; LINES STARTING WITH ';' ARE COMMENTS
title      = Minimization    ; Title of run

; Parameters describing what to do, when to stop and what to save
integrator  = steep          ; Algorithm (steep = steepest descent minimization)
emtol       = 1000.0         ; Stop minimization when the maximum force < 10.0 kJ/mol
emstep      = 0.01           ; Energy step size
nsteps      = 50000          ; Maximum number of (minimization) steps to perform

; Parameters describing how to find the neighbors of each atom and how to calculate the interactions
nstlist     = 1              ; Frequency to update the neighbor list and long range forces
cutoff-scheme = Verlet
ns_type     = grid           ; Method to determine neighbor list (simple, grid)
rlist       = 1.0            ; Cut-off for making neighbor list (short range forces)
coulombtype  = cutoff        ; Treatment of long range electrostatic interactions
rcoulomb     = 1.0           ; long range electrostatic cut-off
rvdw         = 1.0           ; long range Van der Waals cut-off
pbc         = xyz            ; Periodic Boundary Conditions
```

## Energy Minimization

---



接下来要对加完离子的体系做能量最小化了

```
gmx grompp -f em.mdp -c ions.gro -p topol.top -o em.tpr
```

```
gmx mdrun -v -deffnm em
```

# Energy Minimization



mdp文件和之前没有区别

em.mdp

```
; 能量最小化核心参数
;-----
integrator      = steep ;最陡下降法（适合初步结构优化，稳定但较慢）
; integrator    = cg   ;可选：共轭梯度法（收敛更快，可在steep后切换使用）
emtol          = 10.0  ;最大力收敛阈值（Amber推荐≤10.0 kJ/mol/nm）
nsteps         = 10000 ;最大优化步数（确保足够步数以达emtol）
;-----
; 非键相互作用参数
;-----
cutoff-scheme   = Verlet ;Verlet截断方案（自动管理邻居列表，推荐）
nstlist        = 20    ;邻居列表更新频率
rlist          = 1.0   ;邻居列表截断半径（必须≥rvdw/rcoulomb）
vdwtype        = Cut-off ;Amber标准范德华截断
rvdw           = 1.0   ;范德华截断半径
coulombtype     = pme   ;PME处理长程静电
rcoulomb       = 1.0   ;直接空间静电截断（与rvdw对齐）
;-----
; 约束算法
;-----
constraints     = h-bonds ;仅约束氢键（Amber力场参数化基于此设置）
constraint_algorithm = LINCS ;LINCS算法
```

# NVT Equilibration



接下来要对EM结束的体系进行NVT平衡

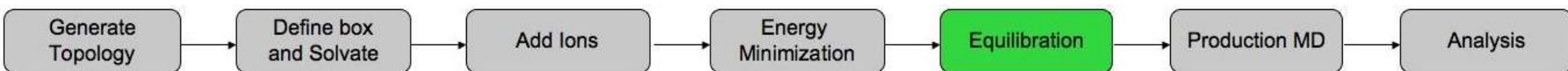
NVT开始就和之前有了一点区别

```
define          = -DPOSRES ;按需启用位置限制
integrator      = md       ;分子动力学积分器
dt              = 0.001    ;时间步长1 fs
nsteps         = 500000    ;500 ps模拟时间
nstxtcout      = 5000     ;每5 ps输出坐标
nstvout        = 5000     ;每5 ps输出速度
nstfout        = 5000     ;每5 ps输出力
nstcalcenergy  = 100      ;每100步计算能量
nstenergy      = 1000     ;每1 ps输出能量
nstlog         = 1000     ;每1 ps输出日志
;非键相互作用
cutoff-scheme  = Verlet    ;Verlet截断方案
nstlist        = 20       ;邻居列表更新频率
rlist          = 1.0      ;邻居列表半径
coulombtype    = pme       ;PME处理静电
rcoulomb       = 1.0      ;静电截断1.0 nm
vdwtype        = Cut-off   ;范德华截断
vdw-modifier   = None
rvdw           = 1.0      ;范德华截断1.0 nm
;温度控制
tcoupl         = V-rescale ;V-rescale控温
tc_grps        = Protein_MOL Water_and_ions;分组控温
tau_t          = 0.2 0.2  ;V-rescale的时间常数
ref_t          = 310.0 310.0 ;参考温度310 K
```

主要区别在于，我们现在多了配体分子，此时在控制体系温度和移除质心平移的时候，我们一般把配体和蛋白质放在同一组

```
;约束参数
constraints    = h-bonds ;约束
constraint_algorithm = LINCS ;LINCS算法
;质心运动移除
nstcomm        = 100     ;每100步移除质心运动
comm_mode      = linear  ;线性模式
comm_grps      = Protein_MOL Water_and_ions;质心移除
;初始速度
gen-vel        = yes     ;生成初始速度
gen-temp       = 310.0   ;初始温度300 K
gen-seed       = -1      ;随机种子
;参考坐标
refcoord_scaling = com    ;基于质心缩放
```

# NVT Equilibration



我们需要准备一个index.ndx文件来告诉gromacs哪些是蛋白和配体原子

```
gmx make_ndx -f em.gro
```

```
0 System : 104760 atoms
1 Protein : 6443 atoms
2 Protein-H : 3224 atoms
3 C-alpha : 408 atoms
4 Backbone : 1224 atoms
5 MainChain : 1634 atoms
6 MainChain+Cb : 2014 atoms
7 MainChain+H : 2026 atoms
8 SideChain : 4417 atoms
9 SideChain-H : 1590 atoms
10 Prot-Masses : 6443 atoms
11 non-Protein : 98317 atoms
12 Other : 56 atoms
13 LAB : 56 atoms
14 NA : 103 atoms
15 CL : 97 atoms
16 Water : 98061 atoms
17 SOL : 98061 atoms
18 non-Water : 6699 atoms
19 Ion : 200 atoms
20 Water and ions : 98261 atoms

nr : group      '!' : not      'name' nr name      'split' nr      Enter: list groups
'a' : atom      '&' : and      'del' nr      'splitres' nr    'l' : list residues
't' : atom type '|' : or      'keep' nr      'splitat' nr     'h' : help
'r' : residue   'r' : nr      'chain' char
'name' : group  'case' : case sensitive
'ri' : residue index

> 1 | 13

Copied index group 1 'Protein'
Copied index group 13 'LAB'
Merged two groups with OR: 6443 56 -> 6499

> name 21 Protein_MOL

> q
```

通过make\_ndx命令的输出，在我的这个例子里，我们看到

- 组1是蛋白
- 组13是我们的小分子
- 组20是剩下的水和离子

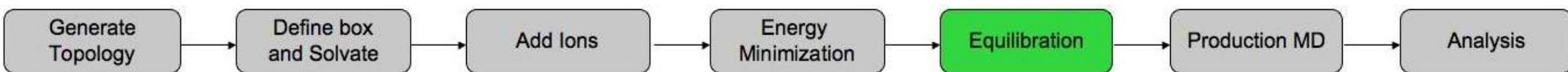
通过 or 逻辑符，我们把蛋白和小分子做成一个新的组，由于已有的组号最多到20，所以新的组会是21

我们把这个新组命名为Protein\_MOL，和mdp文件中保持一致

输入q保存并退出，此时会得到一个index.ndx文件，记录了分组



## NVT Equilibration



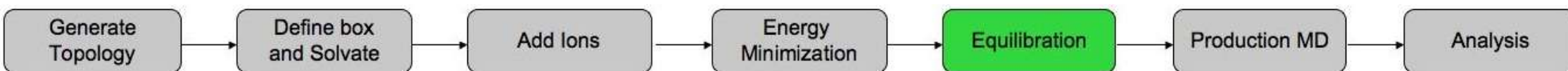
接下来要对EM结束的体系进行NVT平衡

在grompp的时候，把index.ndx提供给gromacs，让它知道nvt.mdp里Protein\_MOL组到底由哪些原子组成

```
gmx grompp -f nvt.mdp -c em.gro -p topol.top -r em.gro -o nvt.tpr -n index.ndx
```

```
gmx mdrun -v -deffnm nvt
```

# NPT Equilibration



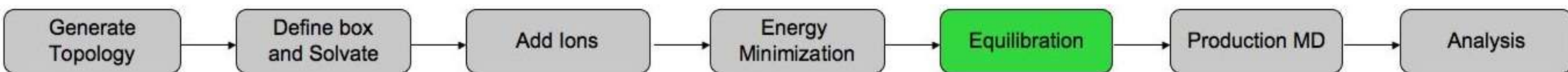
npt.mdp 也更新了分组方式

```
define          = -DPOSRES ; 启用位置限制 (按需调整)
integrator       = md      ; 分子动力学积分器
dt              = 0.002    ; 时间步长2 fs (需约束所有键)
nsteps          = 1000000  ; 2000 ps模拟时间
nstxtcout       = 5000    ; 每10 ps输出坐标
nstvout         = 5000    ; 每10 ps输出速度
nstfout         = 5000    ; 每10 ps输出力
nstcalcenergy   = 100     ; 每100步计算能量
nstenergy       = 1000    ; 每2 ps输出能量
nstlog          = 1000    ; 每2 ps输出日志
; 非键相互作用
cutoff-scheme   = Verlet   ; Verlet截断方案
nstlist         = 20      ; 优化邻居列表更新频率
rlist           = 1.0     ; 邻居列表半径
coulombtype     = pme      ; PME处理静电
rcoulomb        = 1.0     ; 静电截断1.0 nm
vdwtype         = Cut-off  ; 范德华截断
vdw-modifier    = None
rvdw            = 1.0     ; 范德华截断1.0 nm
; 温度控制
tcoupl          = V-rescale ; V-rescale控温
tc_grps         = Protein_MOL Water_and_ions; 分组控温
tau_t           = 0.2 0.2 ; V-rescale的时间常数
ref_t           = 310.0 310.0 ; 参考温度310 K
```

```
; 压力控制
pcoupl          = C-rescale ; C-rescale控压
pcoupltype      = isotropic ; 各向同性压力
tau_p           = 0.5     ; 压力耦合时间常数
compressibility = 4.5e-5   ; 压缩率 (水体系常用4.5e-5 bar-1)
ref_p           = 1.0     ; 参考压力1.0 bar
; 约束参数
constraints     = h-bonds ; 约束
constraint_algorithm = LINCS ; LINCS算法
; 质心运动移除
nstcomm         = 100     ; 每100步移除质心运动
comm_mode       = linear  ; 线性模式
comm_grps       = Protein_MOL Water_and_ions;
; 参考坐标
refcoord_scaling = com    ; 基于质心缩放
```

## NPT Equilibration

---

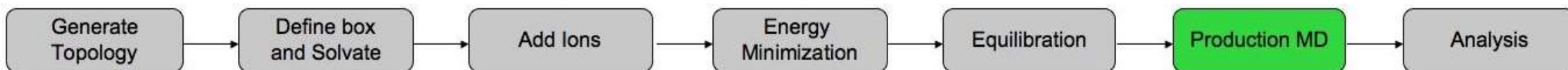


在grompp的时候，把index.ndx提供给gromacs

```
gmx grompp -f npt.mdp -c nvt.gro -p topol.top -r nvt.gro -o npt.tpr -n index.ndx
```

```
gmx mdrun -v -deffnm npt
```

# Production MD



md.mdp      更新了分组方式

```
integrator = md ; leap-frog积分器
nsteps = 250000000 ; 500 ns模拟时间
dt = 0.002 ; 时间步长2 fs
nstxout = 50000 ; trr输出, 设置为0则不输出trr
nstvout = 50000 ;
nstfout = 50000 ;
nstxout-compressed = 50000 ; 每100 ps输出压缩轨迹
compressed-x-precision = 1000 ; 轨迹精度
compressed-x-grps = Protein_MOL ; xtc格式输出组
nstlog = 5000 ; 每10 ps输出日志
nstenergy = 5000 ; 每10 ps输出能量
nstcalcenergy = 100 ;
; 非键相互作用
cutoff-scheme = Verlet
nstlist = 20
rlist = 1.0
coulombtype = pme
rcoulomb = 1.0
vdwtype = Cut-off
vdw-modifier = None
rvdw = 1.0
```

```
; 温度控制
tcoupl = Nose-Hoover ;
tc_grps = Protein_MOL Water_and_ions; 分组控温
tau_t = 1.0 1.0 ;
ref_t = 310.0 310.0 ; 参考温度310 K
; 压力控制
pcoupl = Parrinello-Rahman
pcoupltype = isotropic
tau_p = 5.0 ;
compressibility = 4.5e-5
ref_p = 1.0
; 约束参数
constraints = h-bonds ; 约束
constraint_algorithm = LINCS
continuation = yes
; 质心运动移除
nstcomm = 100 ; 每100步移除质心运动
comm_mode = linear
comm_grps = Protein_MOL Water_and_ions
; 参考坐标
refcoord_scaling = com
```

## Production MD

---



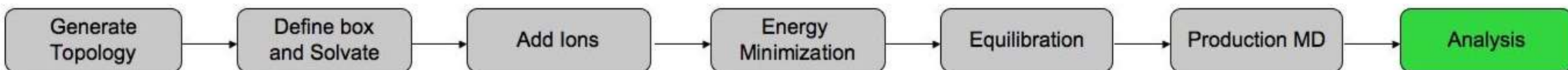
在grompp的时候，把index.ndx提供给gromacs

```
gmx grompp -f md.mdp -c npt.gro -p topol.top -r -o md.tpr -n index.ndx
```

```
gmx mdrun -v -deffnm md -nt 16
```



# Analysis



轨迹后处理，去除周期性边界条件

```
gmx trjconv -f md.xtc -s md.tpr -o md_nopbc.xtc -pbc mol -ur compact -center -n index.ndx
```

消除整体平动和转动

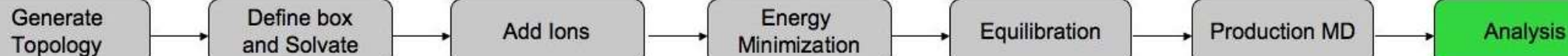
```
gmx trjconv -f md_nopbc.xtc -s md.tpr -o fit.xtc -fit rot+trans -n index.ndx
```

```
Group 21 (Protein_MOL) has 6499 elements
Select a group: 21
Selected 21: 'Protein_MOL'
Select group for output
Group 0 (System) has 104760 elements
```

注意我们这次是有配体的，所以记得选Protein\_MOL组

做完处理之后就可以用VMD看轨迹了

# Ligand RMSD



对去除了周期性边界条件的轨迹

```
gmx trjconv -f md.xtc -s md.tpr -o md_nopbc.xtc -pbc mol -ur compact -center -n index.ndx
```

计算配体的RMSD，这是用来看配体稳定性的必做图

```
gmx rms -s md.tpr -f md_noPBC.xtc -o rmsd.xvg -tu ns -n index.ndx
```

用蛋白对齐每一帧

计算配体的RMSD

```
Select group for least squares fit
Group 0 ( System) has 104760 elements
Group 1 ( Protein) has 6443 elements
Group 2 ( Protein-H) has 3224 elements
Group 3 ( C-alpha) has 408 elements
Group 4 ( Backbone) has 1224 elements
Group 5 ( MainChain) has 1634 elements
Group 6 ( MainChain+Cb) has 2014 elements
Group 7 ( MainChain+H) has 2026 elements
Group 8 ( SideChain) has 4417 elements
Group 9 ( SideChain-H) has 1590 elements
Group 10 ( Prot-Masses) has 6443 elements
Group 11 ( non-Protein) has 98317 elements
Group 12 ( Other) has 56 elements
Group 13 ( LAB) has 56 elements
Group 14 ( NA) has 103 elements
Group 15 ( CL) has 97 elements
Group 16 ( Water) has 98061 elements
Group 17 ( SOL) has 98061 elements
Group 18 ( non-Water) has 6699 elements
Group 19 ( Ion) has 200 elements
Group 20 ( Water_and_ions) has 98261 elements
Group 21 ( Protein_MOL) has 6499 elements
Select a group: 1
Selected 1: 'Protein'
```

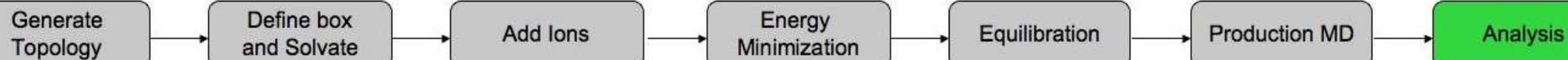
```
Select group for RMSD calculation
Group 0 ( System) has 104760 elements
Group 1 ( Protein) has 6443 elements
Group 2 ( Protein-H) has 3224 elements
Group 3 ( C-alpha) has 408 elements
Group 4 ( Backbone) has 1224 elements
Group 5 ( MainChain) has 1634 elements
Group 6 ( MainChain+Cb) has 2014 elements
Group 7 ( MainChain+H) has 2026 elements
Group 8 ( SideChain) has 4417 elements
Group 9 ( SideChain-H) has 1590 elements
Group 10 ( Prot-Masses) has 6443 elements
Group 11 ( non-Protein) has 98317 elements
Group 12 ( Other) has 56 elements
Group 13 ( LAB) has 56 elements
Group 14 ( NA) has 103 elements
Group 15 ( CL) has 97 elements
Group 16 ( Water) has 98061 elements
Group 17 ( SOL) has 98061 elements
Group 18 ( non-Water) has 6699 elements
Group 19 ( Ion) has 200 elements
Group 20 ( Water_and_ions) has 98261 elements
Group 21 ( Protein_MOL) has 6499 elements
Select a group: 13
Selected 13: 'LAB'
```

得到rmsd.xvg文件，记录了配体rmsd随时间的变化

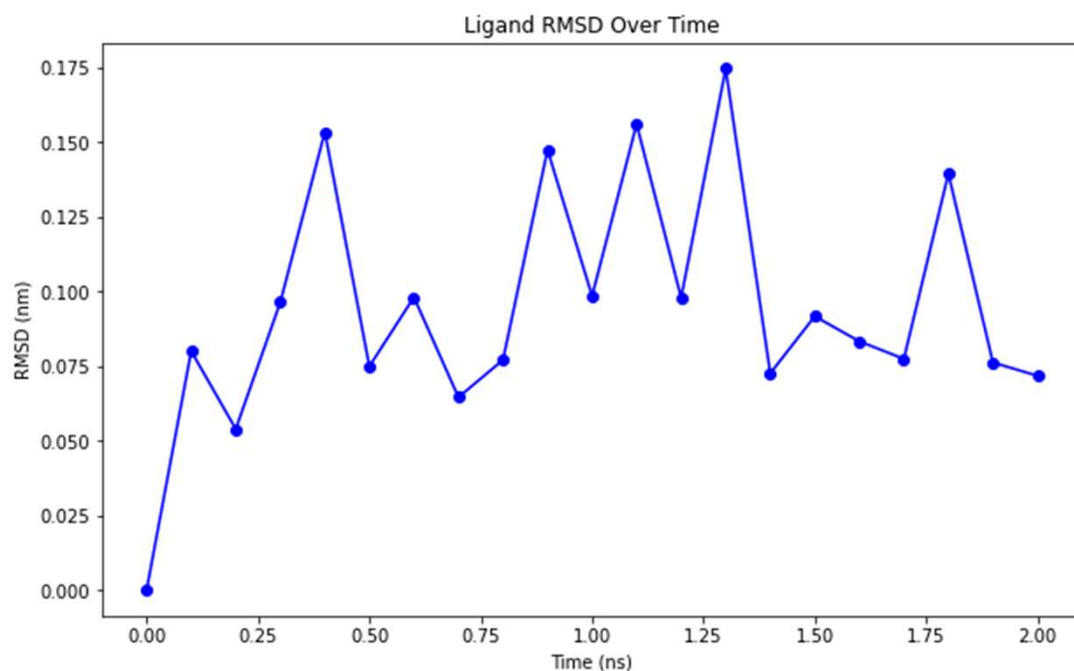


```
## This file was created Sun Mar 2 00:04:49 2025
# Created by:
#
# Executable: /root/software/gromacs-2024.3/bin/gmx
# Data prefix: /root/software/gromacs-2024.3
# Working dir: /root/complex_md
# Command line:
# gmx rms -s md.tpr -f md_noPBC.xtc -o rmsd.xvg -tu ns -n index.ndx
# gmx rms is part of G R O M A C S:
#
# Gys R0wers Mature At Cryogenic Speed
#
@ title "RMSD"
@ xaxis label "Time (ns)"
@ yaxis label "RMSD (nm)"
@TYPE xy
@ subtitle "LAB after lsq fit to Protein_MOL"
0.000000 0.0000120
0.100000 0.0798605
0.200000 0.0538087
0.300000 0.0962225
0.400000 0.1532366
0.500000 0.0747688
0.600000 0.0979257
0.700000 0.0644348
0.800000 0.0769263
0.900000 0.1472737
1.000000 0.0983343
1.100000 0.1558802
1.200000 0.0977604
1.300000 0.1744500
1.400000 0.0724344
1.500000 0.0915253
1.600000 0.0831541
1.700000 0.0772736
1.800000 0.1392832
1.900000 0.0760561
2.000000 0.0716089
```

# Ligand RMSD



得到rmsd.xvg以后作图，可以安装一个xmgrace或者grace打开rmsd.xvg，也可以和我一样写一个python脚本



```
import matplotlib.pyplot as plt
```

```
f = open('rmsd.xvg').readlines()
```

```
time = []
```

```
rmsd = []
```

```
for line in f:
```

```
    if line[0] == '#' or line[0] == '@':
```

```
        continue
```

```
    time_, rmsd_ = line.split()
```

```
    time.append(float(time_))
```

```
    rmsd.append(float(rmsd_))
```

```
# Plot the RMSD over time
```

```
plt.figure(figsize=(10, 6))
```

```
plt.plot(time, rmsd, marker='o', linestyle='-', color='blue')
```

```
plt.title('Ligand RMSD Over Time')
```

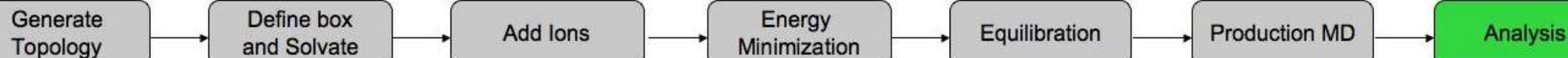
```
plt.xlabel('Time (ns)')
```

```
plt.ylabel('RMSD (nm)')
```

```
plt.savefig('rmsd_plot.png', dpi=300)
```

```
plt.show()
```

# Protein RMSD



对去除了周期性边界条件的轨迹

```
gmx trjconv -f md.xtc -s md.tpr -o md_nopbc.xtc -pbc mol -ur compact -center -n index.ndx
```

计算蛋白的rmsd，一般只算Backbone或者C-alpha

```
gmx rms -s md.tpr -f md_noPBC.xtc -o rmsd.xvg -tu ns -n index.ndx
```

用蛋白对齐每一帧

计算蛋白Backbone或者  
C-alpha的RMSD

得到rmsd.xvg文件，然  
后一样作图即可

```
Select group for least squares fit
Group 0 (System) has 104760 elements
Group 1 (Protein) has 6443 elements
Group 2 (Protein-H) has 3224 elements
Group 3 (C-alpha) has 408 elements
Group 4 (Backbone) has 1224 elements
Group 5 (MainChain) has 1634 elements
Group 6 (MainChain+Cb) has 2014 elements
Group 7 (MainChain+H) has 2026 elements
Group 8 (SideChain) has 4417 elements
Group 9 (SideChain-H) has 1590 elements
Group 10 (Prot-Masses) has 6443 elements
Group 11 (non-Protein) has 98317 elements
Group 12 (Other) has 56 elements
Group 13 (LAB) has 56 elements
Group 14 (NA) has 103 elements
Group 15 (CL) has 97 elements
Group 16 (Water) has 98061 elements
Group 17 (SOL) has 98061 elements
Group 18 (non-Water) has 6699 elements
Group 19 (Ion) has 200 elements
Group 20 (Water_and_ions) has 98261 elements
Group 21 (Protein_MOL) has 6499 elements
Select a group: 1
Selected 1: 'Protein'
```

```
Select group for RMSD calculation
Group 0 (System) has 104760 elements
Group 1 (Protein) has 6443 elements
Group 2 (Protein-H) has 3224 elements
Group 3 (C-alpha) has 408 elements
Group 4 (Backbone) has 1224 elements
Group 5 (MainChain) has 1634 elements
Group 6 (MainChain+Cb) has 2014 elements
Group 7 (MainChain+H) has 2026 elements
Group 8 (SideChain) has 4417 elements
Group 9 (SideChain-H) has 1590 elements
Group 10 (Prot-Masses) has 6443 elements
Group 11 (non-Protein) has 98317 elements
Group 12 (Other) has 56 elements
Group 13 (LAB) has 56 elements
Group 14 (NA) has 103 elements
Group 15 (CL) has 97 elements
Group 16 (Water) has 98061 elements
Group 17 (SOL) has 98061 elements
Group 18 (non-Water) has 6699 elements
Group 19 (Ion) has 200 elements
Group 20 (Water_and_ions) has 98261 elements
Group 21 (Protein_MOL) has 6499 elements
```

