**新流程  
（注意：红色字体为Linux命令，黑色字体为对应的文件内容）**

前期准备  
a.特定模型总原子个数及元素筛选  
#!/usr/bin/env python

# coding: utf-8

# In[13]:

import os

input\_dir = r'D:\dsgdb9nsd.xyz\dsgdb9nsd.xyz'  # 输入目录

output\_dir = r'D:\dsgdb9nsd\_new.xyz'  # 输出目录

# 创建输出目录

if not os.path.exists(output\_dir):

    os.makedirs(output\_dir)

atom\_counts = {'C': 0, 'H': 0, 'O': 0, 'N': 0}  # 统计元素出现次数的字典

def process\_file(file\_path):

    with open(file\_path, 'r') as file:

        # 读取第一行数据

        first\_line = file.readline().strip()

        # 提取数值

        values = [float(x) for x in first\_line.split() if x.isdigit()]

        # 判断数值是否在20到30之间

        if len(values) > 0 and 20 <= values[0] <= 30:

            output\_file\_path = os.path.join(output\_dir, os.path.basename(file\_path))

            with open(output\_file\_path, 'w') as output\_file:

                output\_file.write(first\_line + '\n')

                for line in file:

                    output\_file.write(line)

                print(f'文件 {output\_file\_path} 保存完成')

            for line in open(output\_file\_path, 'r'):

                line = line.strip().split()

                if len(line) > 0 and line[0] in atom\_counts:

                    atom\_counts[line[0]] += 1

# 遍历输入目录下的所有.xyz文件

for file\_name in os.listdir(input\_dir):

    if file\_name.endswith('.xyz'):

        file\_path = os.path.join(input\_dir, file\_name)

        process\_file(file\_path)

# 生成输出文件路径

output\_count\_file\_path = os.path.join(output\_dir, 'atom\_counts.txt')

# 保存统计结果到输出文件

with open(output\_count\_file\_path, 'w') as output\_count\_file:

    for atom, count in atom\_counts.items():

        output\_count\_file.write(f'{atom}: {count}\n')

print('处理完成！')

# In[17]:

import os

import random

import shutil

input\_dir = r'D:\dsgdb9nsd\_new.xyz'  # 输入目录

output\_dir = r'D:\dsgdb9nsd\_1000.xyz'  # 输出目录

# 创建输出目录

if not os.path.exists(output\_dir):

    os.makedirs(output\_dir)

file\_list = [file for file in os.listdir(input\_dir) if file.endswith('.xyz')]

random\_files = random.sample(file\_list, 1000)  # 从文件列表中随机选择 1000 个文件

for file\_name in random\_files:

    file\_path = os.path.join(input\_dir, file\_name)

    output\_file\_path = os.path.join(output\_dir, file\_name)

    shutil.copyfile(file\_path, output\_file\_path)

    print(f'文件 {output\_file\_path} 保存完成')

print('处理完成！')

# In[40]:

import os

import random

import shutil

def count\_atoms(filepath):

    """读取文件并统计关键字出现的次数之和"""

    total\_count = 0

    with open(filepath) as f:

        for line in f:

            atom = line.split()[0]

            if atom in ['C', 'H', 'O', 'N']:

                total\_count += 1

    return total\_count

if \_\_name\_\_ == '\_\_main\_\_':

    directory = r'D:\dsgdb9nsd.xyz'

    output\_directory = r'D:\1000.xyz'

    # 创建保存结果的目录

    os.makedirs(output\_directory, exist\_ok=True)

    # 遍历目录下所有的 .xyz 文件

    files = [f for f in os.listdir(directory) if f.endswith('.xyz')]

    selected\_files = []

    for filename in files:

        # 读取文件

        filepath = os.path.join(directory, filename)

        total\_count = count\_atoms(filepath)

        # 检查出现次数之和是否在20到30之间

        if 20 <= total\_count <= 30:

            selected\_files.append(filepath)

    # 如果筛选的文件数量超过1000个，随机选择1000个文件

    if len(selected\_files) > 1000:

        selected\_files = random.sample(selected\_files, 1000)

    # 将符合条件的文件复制到输出目录

    for i, filepath in enumerate(selected\_files):

        output\_filepath = os.path.join(output\_directory, f'{i+1}.xyz')

        shutil.copy(filepath, output\_filepath)

    print('筛选完成！')

# In[41]:

import os

import random

import shutil

def count\_atoms(filepath):

    """读取文件并统计关键字出现的次数之和"""

    total\_count = 0

    with open(filepath) as f:

        for line in f:

            atom = line.split()[0]

            if atom in ['C', 'H', 'O', 'N']:

                total\_count += 1

    return total\_count

if \_\_name\_\_ == '\_\_main\_\_':

    directory = r'D:\dsgdb9nsd.xyz'

    output\_directory = r'D:\1000\_new.xyz'

    # 创建保存结果的目录

    os.makedirs(output\_directory, exist\_ok=True)

    # 遍历目录下所有的 .xyz 文件

    files = [f for f in os.listdir(directory) if f.endswith('.xyz')]

    selected\_files = []

    for filename in files:

        # 读取文件

        filepath = os.path.join(directory, filename)

        total\_count = count\_atoms(filepath)

        # 检查出现次数之和是否在20到30之间

        if 20 <= total\_count <= 30:

            selected\_files.append(filepath)

    # 如果筛选的文件数量超过1000个，随机选择1000个文件

    if len(selected\_files) > 1000:

        selected\_files = random.sample(selected\_files, 1000)

    # 将符合条件的文件复制到输出目录（保持原有文件名不变）

    for filepath in selected\_files:

        output\_filepath = os.path.join(output\_directory, os.path.basename(filepath))

        shutil.copy(filepath, output\_filepath)

print('筛选完成！')

b.文件格式转换（非标准.xyz>.xyz>.pdb）  
#单个.xyz转成标准的.xyz

def convert\_to\_xyz(input\_file\_path, output\_file\_path):

    with open(input\_file\_path, 'r') as input\_file:

        lines = input\_file.readlines()

# 保存第二行内容

    second\_line = lines[1].strip()

    # 删除第一行

    lines = lines[1:]

    # 提取原子坐标信息，跳过包含数字的行

    atom\_coords = []

    for line in lines:

        line = line.strip().split('\t')

        if len(line) >= 4 and not any(char.isdigit() for char in line[0]):

            atom\_coords.append([line[0]] + [float(coord) for coord in line[1:4]])

    with open(output\_file\_path, 'w') as output\_file:

        # 写入原子数

        num\_atoms = len(atom\_coords)

        output\_file.write(str(num\_atoms) + '\n')

 # 写入第二行内容

        output\_file.write(second\_line + '\n')

        # 写入原子标识符和坐标

        for atom in atom\_coords:

            output\_file.write(f'{atom[0]:2s}\t{atom[1]:14.10f}\t{atom[2]:14.10f}\t{atom[3]:14.10f}\n')

    print(f'Successfully converted {input\_file\_path} to {output\_file\_path}.')

input\_file\_path = r"E:\protein\mole\1000\1000.xyz\916.xyz"

output\_file\_path = r"E:\protein\mole\1000\new.xyz\916.xyz"

convert\_to\_xyz(input\_file\_path, output\_file\_path)

#目录下的所有.xyz转成标准的.xyz

import os

def process\_xyz\_files(input\_dir, output\_dir):

    # 获取输入目录下的所有.xyz文件

    file\_list = [file\_name for file\_name in os.listdir(input\_dir) if file\_name.endswith(".xyz")]

    for file\_name in file\_list:

        input\_file\_path = os.path.join(input\_dir, file\_name)

        output\_file\_path = os.path.join(output\_dir, file\_name)

        convert\_to\_xyz(input\_file\_path, output\_file\_path)

input\_dir = r"E:\protein\mole\1000\1000.xyz"

output\_dir = r"E:\protein\mole\1000\new.xyz"

process\_xyz\_files(input\_dir, output\_dir)

#目录下的所有.xyz转成标准的.xyz

import os

def process\_xyz\_files(input\_dir, output\_dir):

    # 获取输入目录下的所有.xyz文件

    file\_list = [file\_name for file\_name in os.listdir(input\_dir) if file\_name.endswith(".xyz")]

    for file\_name in file\_list:

        input\_file\_path = os.path.join(input\_dir, file\_name)

        output\_file\_path = os.path.join(output\_dir, file\_name)

        convert\_to\_xyz(input\_file\_path, output\_file\_path)

input\_dir = r"E:\protein\mole\1000\1000.xyz"

output\_dir = r"E:\protein\mole\1000\new.xyz"

process\_xyz\_files(input\_dir, output\_dir)

#目录下所有.xyz文件转化为.pdb文件

import os

from openbabel import openbabel as ob

def convert\_xyz\_to\_pdb(xyz\_file, pdb\_file):

    conv = ob.OBConversion()

    conv.SetInFormat('xyz')

    conv.SetOutFormat('pdb')

    mol = ob.OBMol()

    conv.ReadFile(mol, xyz\_file)

    conv.WriteFile(mol, pdb\_file)

def convert\_all\_xyz\_to\_pdb(input\_dir, output\_dir):

    # 确保输出目录存在

    if not os.path.exists(output\_dir):

        os.makedirs(output\_dir)

    # 处理输入目录下的所有.xyz文件

    for root, dirs, files in os.walk(input\_dir):

        for file in files:

            if file.endswith('.xyz'):

                xyz\_file = os.path.join(root, file)

                pdb\_file = os.path.join(output\_dir, f"{os.path.splitext(file)[0]}.pdb")

                convert\_xyz\_to\_pdb(xyz\_file, pdb\_file)

# 调用示例

input\_dir = r"E:\protein\mole\1000\1000.xyz"

output\_dir = r"E:\protein\mole\1000\new\_pbd"

convert\_all\_xyz\_to\_pdb(input\_dir, output\_dir)

c.生成模型文件夹将需要的如下文件放到对应目录中

图形用户界面, 文本, 应用程序

描述已自动生成  
import os

import shutil

source\_dir = r'E:\protein\mole\1000\me'

target\_dir = r'E:\protein\mole\1000\me1'

# 遍历源目录中的所有文件和文件夹

for item in os.listdir(source\_dir):

    item\_path = os.path.join(source\_dir, item)

    if os.path.isfile(item\_path):

        if item.endswith('.pdb'):

            # 如果是 .pdb 文件，创建与文件名相同的目录并将文件复制到目录中

            pdb\_dir = os.path.join(target\_dir, item[:-4])

            os.makedirs(pdb\_dir, exist\_ok=True)

            shutil.copy2(item\_path, pdb\_dir)

        else:

            # 如果非 .pdb 文件，递归地将文件复制到所有子目录中

            for pdb\_dir in os.listdir(target\_dir):

                pdb\_dir\_path = os.path.join(target\_dir, pdb\_dir)

                if os.path.isdir(pdb\_dir\_path):

                    shutil.copy2(item\_path, pdb\_dir\_path)

print("操作完成!")

计算流程

1）cd /home/taotao/free\_energy/335#进入自己的目录

script.pl#修改334.pdb为对应目录的名称

#!/usr/bin/perl

use strict;

use warnings;

# 获取要替换的文件名

my $replacement = shift;

# 在文件名末尾添加.pdb扩展名如果没有提供

$replacement .= '.pdb' unless $replacement =~ /\.pdb$/;

# 读取job1.sh文件

my @lines;

{

    open(my $fh, '<', 'job1.sh') or die "无法打开 job1.sh 文件: $!";

    @lines = <$fh>;

    close($fh);

}

# 替换文件名

foreach my $line (@lines) {

    # 只替换以"334.pdb"结尾的行的文件名

    if ($line =~ /^(.\*?334\.pdb)\b/) {

        $line =~ s/334\.pdb/$replacement/g;

    }

}

# 将修改后的脚本写回job1.sh文件

{

    open(my $fh, '>', 'job1.sh') or die "无法写入 job1.sh 文件: $!";

    print $fh @lines;

    close($fh);

}

# 输出替换完成的消息

print ".pdb 替换为 $replacement\n";

2) perl script.pl

job1.sh#运行如下命令

#!/bin/bash

GMX=/usr/local/gromacs/bin

$GMX/gmx editconf -f 335.pdb -o box.gro -bt dodecahedron -d 1.2 -box 5 5 5

$GMX/gmx solvate -cp box.gro -cs spc216.gro -o solvated.gro -p topol.top -maxsol 600

$GMX/gmx grompp -f em.mdp -c solvated.gro -p topol.top -o em.tpr

$GMX/gmx mdrun -v -deffnm em >& em.txt &

$GMX/gmx grompp -f equil.mdp -c em.gro -p topol.top -o equil.tpr

$GMX/gmx mdrun -deffnm equil >& equil.txt &

3) perl job1.sh

4) python3 /home/taotao/free\_energy/334/lambda.py -d 335

lambda.py#生成lambda目录、复制原目录的top、gro、mdp(mdout和run)文件，并更改run.mdp文件（-d 335为output\_files = '/home/taotao/free\_energy'的补充，补充后为：output\_files = '/home/taotao/free\_energy/335'

）

import os

import shutil

import sys

run\_mdp\_template = "run\_mdp.mdp"

output\_files = '/home/taotao/free\_energy'

def write\_mdp(mdp\_content, mdp\_filename, output\_directory):

    with open(os.path.join(output\_directory, mdp\_filename), 'w') as f:

        f.write(mdp\_content)

# 读取模板文件内容

with open(run\_mdp\_template, 'r') as f:

    mdp\_template\_content = f.read()

#lambda\_value = [0.0, 0.2, 0.4, 0.6, 0.8, 0.9, 1.0]

lambda\_values = [0,1,2,3,4,5,6]

if len(sys.argv) > 1 and sys.argv[1] == '-d':

    if len(sys.argv) > 2:

        output\_files = os.path.join(output\_files, sys.argv[2])

for lambda\_value in lambda\_values:

    lambda\_number = str(lambda\_value)  # 将小数点替换为下划线

    lambda\_directory = os.path.join(output\_files, f'lambda\_0{lambda\_number}')

    if not os.path.exists(lambda\_directory):

        os.mkdir(lambda\_directory)

    gro\_file = os.path.join(output\_files, 'equil.gro')

    top\_file = os.path.join(output\_files, 'topol.top')

    shutil.copy(gro\_file, os.path.join(lambda\_directory, 'equil.gro'))

    shutil.copy(top\_file, lambda\_directory)

    mdp\_content = mdp\_template\_content.format(lambda\_value)

    # 写入替换后的内容到新的mdp文件中

    write\_mdp(mdp\_content, 'run.mdp', lambda\_directory)

5) job.sh#自动进入lambda文件运行如下命令

#!/bin/bash

# Change to the location of your GROMACS-2018 installation

GMX=/usr/local/gromacs/bin

for (( i=0; i<7; i++ ))

do

    LAMBDA=$i

    # A new directory will be created for each value of lambda and

    # at each step in the workflow for maximum organization.

    cd lambda\_0$LAMBDA

    ##############################

    # ENERGY MINIMIZATION STEEP  #

    ##############################

    echo "Starting minimization for lambda = $LAMBDA..."

       $GMX/gmx grompp -f run.mdp -c equil.gro -r equil.gro -p topol.top

       $gmx mdrun >& log.txt &

        cd ../

       cd lambda\_0$LAMBDA

done

exit;

6）gmx bar -b 100 -f lambda\_0?/dhdl.xvg处理数据得到自由能