#Python 3.7

Contents

find_z_add_hdz

10.5.1 The hydrogenated model is stored in the models_H list

11 Merge

11.1 Merging carbon atoms

11.2 Combined hydrogen atoms

11.3 Adjusting the structure of hydrogen

11.4 Merging carbon and hydrogen after structural layout adjustment is completed

11.5 Output the complete M file gro

12 [File Directory] (catalog_2)

13 RS_VDW

13.1 rotor1 and stator1

13.2 rotor2 and stator2

13.3 rotor3 and stator3

13.4 motor and rotor1

13.5 rotor1 and rotor2

13.6 rotor2 and rotor3

14 四、data file

15 五、input file

16 六、lsf file

17 七、document file Model setting - path

18 八、rm_cp_bsub file

19 All variables

```
import lpynb_importer
import pandas as pd
import numpy as np
from pandas import Series,DataFrame
import matplotlib.pyplot as plt
```

```
from mpl_toolkits.mplot3d import Axes3D
%pylab
# %load_ext nbtutor
import os
import shutil
import re
import time

year = str(datetime.datetime.now().year)
month = str(datetime.datetime.now().month)
day = str(datetime.datetime.now().day)
now_time = year+month+day
now_time

# [固定目录] （F:\_python\python_module）

os.chdir((r"F:\_python_1\python_module").replace(r"\\",r"\\\\"))
catalog_0 = os.getcwd()
catalog_0

# [可变目录] （F:\_python\20190516）

# catalog_1_input = (r"F:\_python\20190420").replace(r"\\",r"\\\\")
# catalog_1_input = (r"F:\_python\20190508").replace(r"\\",r"\\\\")
# catalog_1_input = (r"F:\_python\20190516").replace(r"\\",r"\\\\")

# catalog_1_input = (r"F:\_python\20190517").replace(r"\\",r"\\\\")
catalog_1_input = (r"F:\_python_1\20190616").replace(r"\\",r"\\\\")
```

os.chdir(catalog_1_input)

catalog_1 = os.getcwd() #:获得绝对路径

path = catalog_1

catalog_1

\033[1;31;40m　　# 014578 是显示方式（可选），30-37 是字体颜色，40m-47m 是字体背景颜色；

\033[0m　　# 恢复终端默认颜色，即取消颜色设置；

print ("\033[1;31;40mHello world!\033[0m")

path_CatelogsFiles = os.listdir()　　#:列出 dirname 下的目录和文件

print("\033[1;31;46m 当前程序所在路径：　　%s\033[0m\n 路径下的文件目录：　%s"%(path,path_CatelogsFiles))

输入转子 M*S**--￥￥--￥￥--￥￥--￥￥

curfilerename = input("模型为：M*S%%--￥￥--￥￥--￥￥--￥￥")

curfilerename_temp = curfilerename

curfilerename_span0 = re.search("-",curfilerename_temp).span()[0]

curfilerename_temp1 = curfilerename_temp[:curfilerename_span0+1]

curfilerename_temp2 = curfilerename_temp[curfilerename_span0+1:]

curfilerename_temp3 = "_"+curfilerename_temp2.replace("-","-_").replace("-","_-")+"_"

curfilerename = (curfilerename_temp1 + curfilerename_temp3).split("-")

curfilerename

输入定子 S--%%--%%--%%

```
stators = input("转子 123 的定子分别为:S--%%--%%--%%")

stators_temp =  stators

stators_span0 = re.search("-",stators_temp).span()[0]

stators_temp1 = stators_temp[:stators_span0 + 1]

stators_temp2 = stators_temp[stators_span0+1:]
stators_temp3 = "_"+ stators_temp2.replace("-","-_").replace("-","_-")+"_"

stators = (stators_temp1 + stators_temp3).split("-")
stators

# 电机

#电机
# 扶手椅
motor_55 = "F://_python/Models_two_class/Motor_gro/M_r0.339_55_3.074_260.gro"
motor_66 = "F://_python/Models_two_class/Motor_gro/M_r0.4078_66_3.074_312.gro"
motor_77 = "F://_python/Models_two_class/Motor_gro/M_r0.475_77_3.074_364.gro"
motor_88  = "F://_python/Models_two_class/Motor_gro/M_r0.543_88_3.074_416.gro"
motor_99  = "F://_python/Models_two_class/Motor_gro/M_r0.610_99_3.074_468.gro"
motor_1010                                                                    =
"F://_python/Models_two_class/Motor_gro/M_r0.678_1010_3.074_520.gro"
motor_1111                                                                    =
"F://_python/Models_two_class/Motor_gro/M_r0.746_1111_3.074_572.gro"
motor_1212                                                                    =
"F://_python/Models_two_class/Motor_gro/M_r0.813_1212_3.074_624.gro"
motor_1313                                                                    =
```

"F://_python/Models_two_class/Motor_gro/M_r0.881_1313_3.074_676.gro"

motor_1414 = "F://_python/Models_two_class/Motor_gro/M_r0.949_1414_3.074_728.gro"

motor_1515 = "F://_python/Models_two_class/Motor_gro/M_r1.016_1515_3.074_780.gro"

motor_1616 = "F://_python/Models_two_class/Motor_gro/M_r1.084_1616_3.074_832.gro"

锯齿形

motor_80 = "F://_python/Models_two_class/Motor_gro/M_r0.313_80_2.840_224.gro"

motor_90 = "F://_python/Models_two_class/Motor_gro/M_r0.353_90_2.840_252.gro"

motor_100 = "F://_python/Models_two_class/Motor_gro/M_r0.3925_100_2.840_280.gro"

motor_110 = "F://_python/Models_two_class/Motor_gro/M_r0.432_110_3.053_352.gro"

motor_120 = "F://_python/Models_two_class/Motor_gro/M_r0.4704_120_3.053_384.gro"

motor_130 = "F://_python/Models_two_class/Motor_gro/M_r0.509_130_3.053_416.gro"

motor_140 = "F://_python/Models_two_class/Motor_gro/M_r0.548_140_3.053_448.gro"

motor_150 = "F://_python/Models_two_class/Motor_gro/M_r0.587_150_3.053_480.gro"

motor_160 = "F://_python/Models_two_class/Motor_gro/M_r0.627_160_3.053_512.gro"

motor_170 = "F://_python/Models_two_class/Motor_gro/M_r0.665_170_3.053_544.gro"

motor_180 = "F://_python/Models_two_class/Motor_gro/M_r0.703_180_3.053_576.gro"

motor_190 = "F://_python/Models_two_class/Motor_gro/M_r0.744_190_3.053_608.gro"

motor_200 = "F://_python/Models_two_class/Motor_gro/M_r0.783_200_3.053_640.gro"

motor_210 = "F://_python/Models_two_class/Motor_gro/M_r0.822_210_3.053_672.gro"

motor_220 = "F://_python/Models_two_class/Motor_gro/M_r0.8607_220_3.053_704.gro"

motor_230 = "F://_python/Models_two_class/Motor_gro/M_r0.900_230_3.053_736.gro"

motor_240 = "F://_python/Models_two_class/Motor_gro/M_r0.939_240_3.053_768.gro"

motor_250 = "F://_python/Models_two_class/Motor_gro/M_r0.978_250_3.053_800.gro"

motor_260 = "F://_python/Models_two_class/Motor_gro/M_r1.017_260_3.053_832.gro"

motor_models = [motor_55,motor_66,motor_77,motor_88,motor_99,
motor_1010,motor_1111,motor_1212,motor_1313,
motor_1414,motor_1515,motor_1616,

```
            motor_80,motor_90,motor_100,motor_110,motor_120,motor_130,
            motor_140,motor_150,motor_160,motor_170,motor_180,
            motor_190,motor_200,motor_210,motor_220,motor_230,
            motor_240,motor_250,motor_260
           ]

# 转子

#转子
# 扶手椅
rotor_55 = "F://_python/Models_two_class/Rotor_gro/R_r0.339_55_7.993_660.gro"
rotor_66 = "F://_python/Models_two_class/Rotor_gro/R_r0.4078_66_7.993_792.gro"
rotor_77 = "F://_python/Models_two_class/Rotor_gro/R_r0.475_77_7.993_924.gro"
rotor_88 = "F://_python/Models_two_class/Rotor_gro/R_r0.543_88_7.993_1056.gro"
rotor_99 = "F://_python/Models_two_class/Rotor_gro/R_r0.6104_99_7.993_1188.gro"
rotor_1010 = "F://_python/Models_two_class/Rotor_gro/R_r0.678_1010_7.993_1320.gro"
rotor_1111 = "F://_python/Models_two_class/Rotor_gro/R_r0.746_1111_7.993_1452.gro"
rotor_1212 = "F://_python/Models_two_class/Rotor_gro/R_r0.813_1212_7.993_1584.gro"
rotor_1313 = "F://_python/Models_two_class/Rotor_gro/R_r0.881_1313_7.993_1716.gro"
rotor_1414 = "F://_python/Models_two_class/Rotor_gro/R_r0.9485_1414_7.993_1848.gro"
rotor_1515 = "F://_python/Models_two_class/Rotor_gro/R_r1.0162_1515_7.993_1980.gro"
rotor_1616 = "F://_python/Models_two_class/Rotor_gro/R_r1.084_1616_7.993_2112.gro"
# 锯齿形
rotor_80 = "F://_python/Models_two_class/Rotor_gro/R_r0.313_80_7.952_608.gro"
rotor_90 ="F://_python/Models_two_class/Rotor_gro/R_r0.353_90_7.952_684.gro"
rotor_100 ="F://_python/Models_two_class/Rotor_gro/R_r0.3925_100_7.952_760.gro"
rotor_110 ="F://_python/Models_two_class/Rotor_gro/R_r0.4315_110_7.952_880.gro"
rotor_120 = "F://_python/Models_two_class/Rotor_gro/R_r0.470_120_7.952_912.gro"
```

```
rotor_130 = "F://_python/Models_two_class/Rotor_gro/R_r0.5094_130_8.165_1040.gro"
rotor_140 = "F://_python/Models_two_class/Rotor_gro/R_r0.548_140_8.165_1120.gro"
rotor_150 = "F://_python/Models_two_class/Rotor_gro/R_r0.587_150_8.165_1200.gro"
rotor_160 = "F://_python/Models_two_class/Rotor_gro/R_r0.627_160_8.165_1280.gro"
rotor_170 = "F://_python/Models_two_class/Rotor_gro/R_r0.665_170_8.165_1360.gro"
rotor_180 = "F://_python/Models_two_class/Rotor_gro/R_r0.703_180_8.165_1440.gro"
rotor_190 = "F://_python/Models_two_class/Rotor_gro/R_r0.744_190_8.165_1520.gro"
rotor_200 = "F://_python/Models_two_class/Rotor_gro/R_r0.783_200_8.165_1600.gro"
rotor_210 = "F://_python/Models_two_class/Rotor_gro/R_r0.8216_210_8.165_1680.gro"
rotor_220 = "F://_python/Models_two_class/Rotor_gro/R_r0.8607_220_8.165_1760.gro"
rotor_230 = "F://_python/Models_two_class/Rotor_gro/R_r0.900_230_8.165_1840.gro"
rotor_240 = "F://_python/Models_two_class/Rotor_gro/R_r0.9388_240_8.165_1920.gro"
rotor_250 = "F://_python/Models_two_class/Rotor_gro/R_r0.9778_250_8.165_2000.gro"
rotor_260 = "F://_python/Models_two_class/Rotor_gro/R_r1.017_260_8.165_2080.gro"
rotor_models = [rotor_55,rotor_66,rotor_77,rotor_88,rotor_99,
                rotor_1010,rotor_1111,rotor_1212,rotor_1313,
                rotor_1414,rotor_1515,rotor_1616,
                rotor_80,rotor_90,rotor_100,rotor_110,

rotor_120,rotor_130,rotor_140,rotor_150,rotor_160,rotor_170,rotor_180,rotor_190,rotor_
200,
                rotor_210,rotor_220,rotor_230,rotor_240,rotor_250,rotor_260
                ]


# 定子


#定子
# 扶手椅
stator_88 = "F://_python/Models_two_class/Stator_gro/S_r0.543_88_0.5_96.gro"
stator_99 = "F://_python/Models_two_class/Stator_gro/S_r0.605_99_0.5_108.gro"
```

```
stator_1010 = "F://_python/Models_two_class/Stator_gro/S_r0.678_1010_0.5_120.gro"
stator_1111 = "F://_python/Models_two_class/Stator_gro/S_r0.746_1111_0.5_132.gro"
stator_1212 = "F://_python/Models_two_class/Stator_gro/S_r0.813_1212_0.5_144.gro"
stator_1313 = "F://_python/Models_two_class/Stator_gro/S_r0.881_1313_0.5_156.gro"
stator_1414 = "F://_python/Models_two_class/Stator_gro/S_r0.9485_1414_0.5_168.gro"
stator_1515 = "F://_python/Models_two_class/Stator_gro/S_r1.017_1515_0.5_180.gro"
stator_1616 = "F://_python/Models_two_class/Stator_gro/S_r1.084_1616_0.5_192.gro"
stator_1717 = "F://_python/Models_two_class/Stator_gro/S_r1.152_1717_0.5_204.gro"
stator_1818 = "F://_python/Models_two_class/Stator_gro/S_r1.221_1818_0.5_216.gro"
stator_1919 = "F://_python/Models_two_class/Stator_gro/S_r1.2868_1919_0.5_228.gro"
stator_2020 = "F://_python/Models_two_class/Stator_gro/S_r1.3546_2020_0.5_240.gro"
stator_2121 = "F://_python/Models_two_class/Stator_gro/S_r1.422_2121_0.5_252.gro"
stator_2222 = "F://_python/Models_two_class/Stator_gro/S_r1.4899_2222_0.5_264.gro"
# 锯齿形
stator_130 = "F://_python/Models_two_class/Stator_gro/S_r0.5094_130_0.5_104.gro"
stator_140 = "F://_python/Models_two_class/Stator_gro/S_r0.548_140_0.5_112.gro"
stator_150 = "F://_python/Models_two_class/Stator_gro/S_r0.5874_150_0.5_120.gro"
stator_160 = "F://_python/Models_two_class/Stator_gro/S_r0.627_160_0.5_128.gro"
stator_170 = "F://_python/Models_two_class/Stator_gro/S_r0.665_170_0.5_136.gro"
stator_180 = "F://_python/Models_two_class/Stator_gro/S_r0.703_180_0.5_144.gro"
stator_190 = "F://_python/Models_two_class/Stator_gro/S_r0.744_190_0.5_152.gro"
stator_200 = "F://_python/Models_two_class/Stator_gro/S_r0.7826_200_0.5_160.gro"
stator_210 = "F://_python/Models_two_class/Stator_gro/S_r0.822_210_0.5_168.gro"
stator_220 = "F://_python/Models_two_class/Stator_gro/S_r0.861_220_0.5_176.gro"
stator_230 = "F://_python/Models_two_class/Stator_gro/S_r0.900_230_0.5_184.gro"
stator_240 = "F://_python/Models_two_class/Stator_gro/S_r0.939_240_0.5_192.gro"
stator_250 = "F://_python/Models_two_class/Stator_gro/S_r0.978_250_0.5_200.gro"
stator_260 = "F://_python/Models_two_class/Stator_gro/S_r1.017_260_0.5_208.gro"
stator_270 = "F://_python/Models_two_class/Stator_gro/S_r1.056_270_0.5_216.gro"
```

```
stator_280 = "F://_python/Models_two_class/Stator_gro/S_r1.0951_280_0.5_224.gro"
stator_290 = "F://_python/Models_two_class/Stator_gro/S_r1.1342_290_0.5_232.gro"
stator_300 = "F://_python/Models_two_class/Stator_gro/S_r1.1732_300_0.5_240.gro"
stator_310 = "F://_python/Models_two_class/Stator_gro/S_r1.2086_310_0.5_248.gro"
stator_320 = "F://_python/Models_two_class/Stator_gro/S_r1.251_320_0.5_256.gro"
stator_330 = "F://_python/Models_two_class/Stator_gro/S_r1.2904_330_0.5_264.gro"
stator_340 = "F://_python/Models_two_class/Stator_gro/S_r1.3295_340_0.5_272.gro"
stator_350 = "F://_python/Models_two_class/Stator_gro/S_r1.369_350_0.5_280.gro"
stator_360 = "F://_python/Models_two_class/Stator_gro/S_r1.4077_360_0.5_288.gro"
stator_models = [stator_88,stator_99,stator_1010,stator_1111,stator_1212,stator_1313,
                 stator_1414,stator_1515,stator_1616,stator_1717,stator_1818,stator_1919,
                 stator_2020,stator_2121,stator_2222,
                 stator_130,stator_140,stator_150,
                 stator_160,stator_170,stator_180,stator_190,stator_200,
                 stator_210,stator_220,

stator_230,stator_240,stator_250,stator_260,stator_270,stator_280,stator_290,stator_300,
                 stator_310,stator_320,stator_330,stator_340,stator_350,stator_360
                 ]

# 一、修改程序文件名 3stageM*--电机--转子 1--转子 2--转子 3

# 二、修改生成的文件名 3stageM*--电机--转子 1--转子 2--转子 3

filename_head = curfilerename_temp
filename_tail = ".gro"

print(filename_head)
```

三、修改模型结构-电机-转子-定子-匹配

#确定 nanomotor 结构

total_type = 20 #10 个碳纳米管 每个碳纳米管都有一个氢原子组 一共 20 个原子类型

amount = int(total_type/2)

#################### M22 ##############################

############### 电机 ###################

p = re.compile(curfilerename[1])

for e in motor_models:

if p.search(e):

print("\033[1;31;47minfile_motor = %s\033[0m",os.path.split(e))

infile_motor =e

############### 转子 ###################

Rotor1

p = re.compile(curfilerename[2])

for e in rotor_models:

if p.search(e):

print("\033[1;32;47minfile_rotor1 = %s\033[0m",os.path.split(e))

infile_rotor1 =e

Rotor2

p = re.compile(curfilerename[3])

for e in rotor_models:

if p.search(e):

print("\033[1;33;47minfile_rotor2 = %s\033[0m",os.path.split(e))

infile_rotor2 =e

Rotor3

```
p = re.compile(curfilerename[4])
for e in rotor_models:
    if p.search(e):
        print("\033[1;34;47minfile_rotor3 = %s\033[0m]",os.path.split(e))
        infile_rotor3 =e

    ################ 定子  ####################
    ################  需要选择 定子子 1
p = re.compile(stators[1])
for e in stator_models:
    if p.search(e):
        print("\033[1;32;47minfile_stator1 = %s\033[0m]",os.path.split(e))
        infile_stator1 =e
infile_stator2 = infile_stator1

################  需要选择 定子 3
p = re.compile(stators[2])
for e in stator_models:
    if p.search(e):
        print("\033[1;33;47minfile_stator3 = %s\033[0m]",os.path.split(e))
        infile_stator3 =e
infile_stator4 = infile_stator3

################  需要选择 定子 5
p = re.compile(stators[3])
for e in stator_models:
    if p.search(e):
        print("\033[1;34;47minfile_stator5 = %s\033[0m]",os.path.split(e))
        infile_stator5 =e
```

```
infile_stator6 = infile_stator5


infile = [infile_motor,infile_rotor1,infile_rotor2,infile_rotor3,
        infile_stator1,infile_stator2,infile_stator3,infile_stator4,infile_stator5,infile_stator6
        ]


## 函数


#函数定义
# %%nbtutor
################################### 定 义 函 数 ###################################################################
################################### gro 文 件 ###################################################################


### A、定义读取文件函数 gro_file


######## A、定义读取文件函数 gro_file ###
# 参数 infile 为包含文件路径和文件名的字符串
def gro_file(infile):
    with open(infile,'r') as f:
        rows = [row for row in f]
    rows = [row.split() for row in rows[2:-1]]    # 不分隔 有的列读不出来 df.loc[df[5].isin([ '0.000','3.474', '11.299',\])]
    return rows


### B、 调整结构布局函数


########  B、 调整结构布局函数  ####################
#五个参数
```

```
# 1、df_X  为调用每个单独的 gro 文件
# 2、sequence_atomic 为 df_X 递增的原子序列
# 3、distance_origin  为每个 CNT 左端到原点的距离
# 4、first_third   为调整第一列 123CNT    第三列 123 的序列
# 5、 first_column 为系统不同部分分配不同的别名  motor rotor1  rotor2 stators
def adjust_gro_format(df_X,sequence_atomic,distance_origin,first_third,first_column):
    # 修改第一列 df_1[0]
    for  i,e in enumerate(sequence_atomic):
        atomic_number = first_third + sequence_atomic[i]
        if first_column == 'motor' :
            df_X.iloc[i][0]=(str(atomic_number)+'CMO')
        elif first_column == 'rotor1' :
            df_X.iloc[i][0]=(str(atomic_number)+'CR1')
        elif first_column == 'rotor2' :
            df_X.iloc[i][0]=(str(atomic_number)+'CR2')
        elif first_column == 'rotor3' :
            df_X.iloc[i][0]=(str(atomic_number)+'CR3')
        elif first_column == 'rotor4' :
            df_X.iloc[i][0]=(str(atomic_number)+'CR4')
        elif first_column == 'stator1' :
            df_X.iloc[i][0]=(str(atomic_number)+'CS1')
        elif first_column == 'stator2' :
            df_X.iloc[i][0]=(str(atomic_number)+'CS2')
        elif first_column == 'stator3' :
            df_X.iloc[i][0]=(str(atomic_number)+'CS3')
        elif first_column == 'stator4' :
            df_X.iloc[i][0]=(str(atomic_number)+'CS4')
        elif first_column == 'stator5' :
            df_X.iloc[i][0]=(str(atomic_number)+'CS5')
```

```
        elif first_column == 'stator6' :
            df_X.iloc[i][0]=(str(atomic_number)+'CS6')
        elif first_column == 'stator7' :
            df_X.iloc[i][0]=(str(atomic_number)+'CS7')
        elif first_column == 'stator8' :
            df_X.iloc[i][0]=(str(atomic_number)+'CS8')
    # 修改第三列 df_1[2]
    for  i,e in enumerate(sequence_atomic):
        atomic_number = first_third + sequence_atomic[i]
        df_X.iloc[i][2]= atomic_number
    #修改 z 坐标 df_1[5]
    df_X[5] = df_X[5].apply(float).map(lambda x: x + distance_origin)
    return df_X

### C、格式化为 VMD 格式函数

########    C、格式化为 VMD 格式函数##
## 格式化为 VMD 格式  存储在 data
# pandasdataframe 格式    原样      输出保存为 gro 格式
# 传入两个参数
# 1、要存储的数据 dataframe
# 2、输出的文件名 outfile
def dataframe_output_gro(dataframe,outfile):
    dataframe[3] = dataframe[3].apply(float)
    dataframe[4] = dataframe[4].apply(float)
    dataframe[5] = dataframe[5].apply(float)
    temp = dataframe
    data = []
    for i in range(temp.shape[0]):
```

```
        entry= temp.iloc[i]
        mol_type = entry[0]
        atom_type = entry[1]
        num = entry[2]
        x =  "{:.3f}".format(temp.iloc[i][3])
        y =  "{:.3f}".format(temp.iloc[i][4])
        z =  "{:.3f}".format(temp.iloc[i][5])
        out_line = mol_type.rjust(8) + atom_type.rjust(5) + str(num).rjust(7) + \
                   x.rjust(8) + y.rjust(8) + z.rjust(8) + '\n'
        data.append(out_line)
    ## 加 VMD 文件头和尾    保存文件 outfile
    start = 'generated by VMD, t= 0.000000\n{}\n'.format(len(data))
    end = '   0.00000   0.00000   0.00000\n'
    with open(outfile,'w') as wf:
        wf.write(start)
        wf.writelines(data)
        wf.write(end)


### D、生成 data 文件


########################################          D 、    data  文 件
#########################################################################
######

########D、原始文件格式化为 data 格式函数#
# 注：位移单位为 埃（0.1 纳米）
#1、rows 为 gro 格式的 DataFrame  一个碳纳米管组件
#2、add_xyz 为每一个碳纳米管左端的偏移量
# 3、clas: 原子类型，如果模型中共有 2 个原子类型，那么 clas 取值为 1 或者 2
# 4、count: 原子 id 起始值，如果 id 从 1 开始，此时 count = 1
def proces_file(rows,add_xyz,clas,count):
```

```
    raw_rows = rows[2:-1]
    str_xyz = [[float(e) for e in (row[20:].split())]for row in raw_rows]
    atoms = [("%8d"% (i+1+count)) + \
             ("%6d"%clas) + \
             ("%9.3f"%(10*e[0]+add_xyz[0]))+ ("%8.3f"%(10*e[1]+add_xyz[1])) + ("%8.3f"%(10*e[2]+add_xyz[2])) + '\n'
            for i,e in enumerate(str_xyz)]
    return atoms


#### E、 data 文件的 head  格式函数


########  E、 data 文件的 head  格式函数  ###
# 1、raw_xyz_range 为盒子的三维范围 列表类型
# 2、atoms_num 为 data 文件总原子数
def label(raw_xyz_range,atoms_num,relative_mass):
    xyz_l_h = ['  xlo xhi',
               '  ylo yhi',
               '  zlo zhi'
              ]
    xyz_range = [("%11.5f"%float(e[0])+"%12.5f"%float(e[1])+xyz_l_h[i]+'\n')
                    for i,e in enumerate(raw_xyz_range)
                ]
    label_1 = ['# Two class nanotubes hydrogenated \n',
           ("%8d"%atoms_num) + '  atoms\n',
            '\n'
           ]
    label_2 = ["%5s"%(total_type) + ' atom types\n'] + xyz_range


    label_3 =['\n',' Masses\n','\n']+ relative_mass.split("_") + ['\n','Atoms\n','\n' ]
```

```
#     del label_3[-4]
    label = label_1 + label_2 + label_3
    return label


### F、 读取 gro 文件


########    F、 读取 gro 文件    ##################
# 读取 gro 文件，逐行以字符串格式存入 rows 中
# rows 为一个 list
def gro_rows(grofile):
    with open(grofile, 'r') as f:
        rows = [row for row in f]
    return pd.DataFrame(rows)
#     return [row for row in f]


### 生成 in 文件


#########################################################          in  文 件
#####################################################################


#### G、  L-J pair_coeff 格式化


########    G、  L-J pair_coeff 格式化   ##########
# label_2     L-J pair_coeff
# serial_number is start
# amount is end
def pair_coeff(type_type,serial_number,amount):
    temp = []
    j = 0
```

```
#####################  C-C  #######
   if type_type == "C-C":
      epslon = 0.002840
      sigama = 3.400
      if serial_number < amount - 1:
         pair_coeff_C = ["pair_coeff" + "%5d"%(serial_number) + "%5d*%-5d"%(serial_number + 1,amount) +  "%10s"%("lj/cut") +\
                  "%10.6f"%(epslon) +  "%10.3f"%(sigama)+"\n"]
      else:
         pair_coeff_C = ["pair_coeff" + "%5d"%(serial_number) + "%8d"%(amount) + "%13s"%("lj/cut") +\
                  "%10.6f"%(epslon) +  "%10.3f"%(sigama)+"\n"]
      temp= temp + pair_coeff_C
#####################  C-H  #########
   elif type_type == "C-H" :
      epslon = 0.001376
      sigama = 3.025
      if serial_number == 1:
         pair_coeff_C = ["pair_coeff" + "%5d"%(serial_number) + "%5d*%-5d"%(amount + serial_number +1 ,2*amount) +  "%10s"%("lj/cut") +\
                  "%10.6f"%(epslon) +  "%10.3f"%(sigama) + "\n"]
      elif serial_number == 2:
         pair_coeff_A = ["pair_coeff" + "%5d"%(serial_number) + "%8d"%(amount + serial_number - 1) +  "%13s"%("lj/cut") +\
                  "%10.6f"%(epslon) +  "%10.3f"%(sigama) + "\n"]
         pair_coeff_B = ["pair_coeff" + "%5d"%(serial_number) + "%5d*%-5d"%(amount + serial_number + 1,2*amount) +  "%10s"%("lj/cut") +\
                  "%10.6f"%(epslon) +  "%10.3f"%(sigama) + "\n"]
         pair_coeff_C = pair_coeff_A + pair_coeff_B
      elif serial_number > 2 and serial_number < amount -1 :
         pair_coeff_A = ["pair_coeff" + "%5d"%(serial_number) + "%5d*%-5d"%(amount +
```

```
1,amount + serial_number -1) +  "%10s"%("lj/cut") +\
                "%10.6f"%(epslon) +  "%10.3f"%(sigama) + "\n"]
            pair_coeff_B = ["pair_coeff" + "%5d"%(serial_number) + "%5d*%-5d"%(amount + serial_number + 1,2*amount) +  "%10s"%("lj/cut") +\
                "%10.6f"%(epslon) +  "%10.3f"%(sigama) + "\n"]
            pair_coeff_C = pair_coeff_A + pair_coeff_B
        elif serial_number ==  amount -1:
            pair_coeff_A = ["pair_coeff" + "%5d"%(serial_number) + "%5d*%-5d"%(amount  + 1,2*amount-2) +  "%10s"%("lj/cut") +\
                "%10.6f"%(epslon) +  "%10.3f"%(sigama) + "\n"]
            pair_coeff_B = ["pair_coeff" + "%5d"%(serial_number) + "%8d"%(2*amount) + "%13s"%("lj/cut") +\
                "%10.6f"%(epslon) +  "%10.3f"%(sigama) + "\n"]
            pair_coeff_C = pair_coeff_A + pair_coeff_B
        else :
            pair_coeff_C = ["pair_coeff" + "%5d"%(serial_number) + "%5d*%-5d"%(amount + 1 ,2*amount-1) +  "%10s"%("lj/cut") +\
                "%10.6f"%(epslon) +  "%10.3f"%(sigama) + "\n"]
        temp= temp + pair_coeff_C
####################  H-H  ##########
    elif type_type == "H-H" :
        epslon = 0.001500
        sigama = 2.650
        if serial_number < amount - 1:
            pair_coeff_C = ["pair_coeff" + "%5d"%(serial_number+amount) + "%5d*%-5d"%(serial_number + 1+ amount, 2*amount) +\
                "%10s"%("lj/cut") + "%10.6f"%(epslon) +  "%10.3f"%(sigama) + "\n"]
        else:
            pair_coeff_C = ["pair_coeff" + "%5d"%(serial_number+amount) + "%8d"%(2*amount) +\
                "%13s"%("lj/cut") + "%10.6f"%(epslon) +  "%10.3f"%(sigama) + "\n"]
```

```
        temp= temp + pair_coeff_C
    return temp

#### H、转子左右两端固定区域函数

#################### H、转子左右两端固定区域函数 ######
def X_region(df_X_ad,model,dz):
    df_X_ad[3] = df_X_ad[3].apply(float).map(lambda x: x )
    df_X_ad[4] = df_X_ad[4].apply(float).map(lambda x: x )
    px = max(df_X_ad.iloc[:][3])*10
    pz_L = df_X_ad.iloc[0][5]*10
    pz_R = df_X_ad.iloc[-1][5]*10
    region_L =  ["region       %s    block"%(model + "_L") + "%10.3f"%(-px) + "%10.3f"%(px)\
                + "%10.3f"%(-px) + "%10.3f"%(px) + "%10.3f"%(pz_L) + "%10.3f"%(pz_L+dz)+"\n"]
    region_R =  ["region       %s    block"%(model + "_R") + "%10.3f"%(-px) + "%10.3f"%(px)\
                + "%10.3f"%(-px) + "%10.3f"%(px) + "%10.3f"%(pz_R-dz) + "%10.3f"%(pz_R)+"\n"]
    return region_L + region_R

#### I、根据原子类型分组

#################### I、根据原子类型分组###############
def group_type(atom,group_type,amount):
    results = []
    if atom == "C":
        for i in range(0,amount):
            results.append("group             " + "%-20s"%(group_type[i]) + "       type     " +
"%3d"%(i+1)+"\n")
#             if i == 1 or i == 2 or i == 3 :
#                 continue
```

```
#            else:
#                    results.append("group        " + "%-20s"%(group_type[i]) + "    type    " + "%3d"%(i+1)+"\n")
    elif atom == "H":
        for i in range(0,amount):   #amount
            if i == 0:
                results.append("group           " + "%-20s"%(group_type[i]) + "      type    " + "%3d"%(i+1+amount)+"\n")
                break
    return results


### 末端氢化


#### A、定义找出端点 z 坐标函数 find_z_add_hdz


#### A、定义找出端点 z 坐标函数 find_z_add_hdz


# df_adjustment 为 7gros 合并后的 pandasdataframe
# tube_direction 为碳纳米管加氢的末端
# direction 为  L or R
# z_coord 为需要氢化的碳纳米管的末端 z 坐标
# hdz 为碳氢的键长
                                    # 苯环 C-C length = C- H =0.1418nm
                                    # 甲烷  length = C- H =0.1087nm
# df_adjustment 为合并后的 gro pandasdataframe 格式的 df_ad  和  三个参数
hdz = 0.089
# hdz = 0.0628 #  M4 定子 1R 上的氢坏掉了
# hdz = 0.1087
def find_z_add_hdz(df_adjustment,direction,hdz):
    if direction == "L":
```

```
        z_coord = df_adjustment.iloc[0][5]
        tube_z_coord = df_adjustment.loc[df_adjustment[5].isin([z_coord])]
        tube_z_coord[5] = tube_z_coord[5].apply(float).map(lambda x:  x - hdz )
    elif direction == "R":
        z_coord = df_adjustment.iloc[-1][5]
        tube_z_coord = df_adjustment.loc[df_adjustment[5].isin([z_coord])]
        tube_z_coord[5] = tube_z_coord[5].apply(float).map(lambda x:  x + hdz )
    elif direction == "LR":
        z_coord_L = df_adjustment.iloc[0][5]
        tube_z_coord_L = df_adjustment.loc[df_adjustment[5].isin([z_coord_L])]
        tube_z_coord_L[5] = tube_z_coord_L[5].apply(float).map(lambda x:  x - hdz )

        z_coord_R = df_adjustment.iloc[-1][5]
        tube_z_coord_R = df_adjustment.loc[df_adjustment[5].isin([z_coord_R])]
        tube_z_coord_R[5] = tube_z_coord_R[5].apply(float).map(lambda x:  x + hdz )
        tube_z_coord = tube_z_coord_L.append(tube_z_coord_R)
    else:
        print("请选择碳纳米管的末端：L or R")
    return tube_z_coord

## A、读入碳纳米管 gro 文件

#A、读入碳纳米管 gro 文件 #######
# pandas dataframe  格式
df_motor = pd.DataFrame(gro_file(infile_motor))   #电机 DataFrame
df_rotor1 = pd.DataFrame(gro_file(infile_rotor1))  #转子氢 DataFrame
df_rotor2 = pd.DataFrame(gro_file(infile_rotor2))
df_rotor3 = pd.DataFrame(gro_file(infile_rotor3))
df_stator1 = pd.DataFrame(gro_file(infile_stator1))   #定子 DataFrame
```

```
df_stator2 = pd.DataFrame(gro_file(infile_stator2))

df_stator3 = pd.DataFrame(gro_file(infile_stator3))

df_stator4 = pd.DataFrame(gro_file(infile_stator4))

df_stator5 = pd.DataFrame(gro_file(infile_stator5))

df_stator6 = pd.DataFrame(gro_file(infile_stator6))



# 结构布局碳纳米管的长度

# 从原点开始，转子 1--》转子 2--》电机--》转子 3--》转子 4--》定子 1--》定子 2--》。。。--》定子 8

L2                                                  =                                                  6.302
##################################################################################################################

# L2 = float(input("请输入 L2 长度："))

rotor1_length = float(df_rotor1.iloc[-1][5]) - float(df_rotor1.iloc[0][5])  #转子的长度

rotor2_length = float(df_rotor2.iloc[-1][5]) - float(df_rotor2.iloc[0][5])

motor_length = float(df_motor.iloc[-1][5]) - float(df_motor.iloc[0][5]) #3.704  电机的长度

rotor3_length = float(df_rotor3.iloc[-1][5]) - float(df_rotor3.iloc[0][5])  #转子的长度



stator1_length = float(df_stator1.iloc[-1][5]) - float(df_stator1.iloc[0][5])  #定子的长度

stator2_length = float(df_stator2.iloc[-1][5]) - float(df_stator2.iloc[0][5])

stator3_length = float(df_stator3.iloc[-1][5]) - float(df_stator3.iloc[0][5])

stator4_length = float(df_stator4.iloc[-1][5]) - float(df_stator4.iloc[0][5])

stator5_length = float(df_stator5.iloc[-1][5]) - float(df_stator5.iloc[0][5])

stator6_length = float(df_stator6.iloc[-1][5]) - float(df_stator6.iloc[0][5])

## B、 调整碳和氢的结构布局
```

#B、 调整碳和氢的结构布局 ###

dz_01 = 0.1

dz_02 = 0.2

dz_03 = 0.3

dz_04 = 0.4

电机

1、motor gro 调整第一列(不同部分分组）， 第三列和 z 坐标

motor_count = df_motor[0].count()+ 1 #本身原子总数

df_X 和 四个参数

sequence_atomic = np.arange(1,motor_count) #本身原子序列

motor_distance_origin = 0 #左端与远点的距离

motor_sequence = 0 #在结构布局里的原子起始值 等于前面的所有碳纳米管原子总数的和

motor_first_column = "motor" #不同部分的第一列不同的赋予不同的 name

df_motor_ad = adjust_gro_format(df_motor,sequence_atomic,motor_distance_origin,motor_sequence,motor_first_column)

####### 转子 ####

2、rotor1 gro 调整第一列(不同部分分组）， 第三列和 z 坐标

rotor1_count = df_rotor1[0].count()+ 1 #本身原子总数

df_X 和 四个参数

sequence_atomic = np.arange(1,rotor1_count) #本身原子序列

rotor1_distance_origin = motor_distance_origin + motor_length + dz_04 #左端与远点的距离

rotor1_sequence = df_motor[0].count() #在结构布局里的原子起始值 等于前面的所有碳纳米管原子总数的和

rotor1_first_column = "rotor1" #不同部分的第一列不同的赋予不同的 name

df_rotor1_ad = adjust_gro_format(df_rotor1,sequence_atomic,rotor1_distance_origin,rotor1_sequence,r

otor1_first_column)

3、rotor2 gro 调整第一列(不同部分分组），第三列和 z 坐标

rotor2_count = df_rotor2[0].count()+ 1 #本身原子总数

df_X 和 四个参数

sequence_atomic = np.arange(1,rotor2_count) #本身原子序列

rotor2_distance_origin = rotor1_distance_origin + rotor1_length + dz_04 #左端与远点的距离

rotor2_sequence = rotor1_sequence + df_rotor1[0].count() #在结构布局里的原子起始值 等于前面的所有碳纳米管原子总数的和

rotor2_first_column = "rotor2" #不同部分的第一列不同的赋予不同的 name

df_rotor2_ad = adjust_gro_format(df_rotor2,sequence_atomic,rotor2_distance_origin,rotor2_sequence,rotor2_first_column)

4、rotor3 gro 调整第一列(不同部分分组），第三列和 z 坐标

rotor3_count = df_rotor3[0].count()+ 1 #本身原子总数

df_X 和 四个参数

sequence_atomic = np.arange(1,rotor3_count) #本身原子序列

rotor3_distance_origin = rotor2_distance_origin + rotor2_length + dz_04 #左端与远点的距离

rotor3_sequence = rotor2_sequence + df_rotor2[0].count() #在结构布局里的原子起始值 等于前面的所有碳纳米管原子总数的和

rotor3_first_column = "rotor3" #不同部分的第一列不同的赋予不同的 name

df_rotor3_ad = adjust_gro_format(df_rotor3,sequence_atomic,rotor3_distance_origin,rotor3_sequence,rotor3_first_column)

######### 定子 ##########

1、stator1 gro 调整第一列(不同部分分组），第三列和 z 坐标

stator1_count = df_stator1[0].count()+ 1 #本身原子总数

df_X 和 四个参数

sequence_atomic = np.arange(1,stator1_count) #本身原子序列

stator1_distance_origin = df_rotor1_ad.iloc[0][5] + dz_04 #左端与远点的距离

stator1_sequence = rotor3_sequence + df_rotor3[0].count() #在结构布局里的原子起始值 等于前面的所有碳纳米管原子总数的和

stator1_first_column = "stator1" #不同部分的第一列不同的赋予不同的 name

df_stator1_ad = adjust_gro_format(df_stator1,sequence_atomic,stator1_distance_origin,stator1_sequence,stator1_first_column)

2、stator2 gro 调整第一列(不同部分分组），第三列和 z 坐标

stator2_count = df_stator2[0].count()+ 1 #本身原子总数

df_X 和 四个参数

sequence_atomic = np.arange(1,stator2_count) #本身原子序列

stator2_distance_origin = df_rotor1_ad.iloc[-1][5] - stator2_length - dz_04 #+ dz_01 #左端与远点的距离

stator2_sequence = stator1_sequence + df_stator1[0].count() #在结构布局里的原子起始值 等于前面的所有碳纳米管原子总数的和

stator2_first_column = "stator2" #不同部分的第一列不同的赋予不同的 name

df_stator2_ad = adjust_gro_format(df_stator2,sequence_atomic,stator2_distance_origin,stator2_sequence,stator2_first_column)

3、stator3 gro 调整第一列(不同部分分组），第三列和 z 坐标

stator3_count = df_stator3[0].count()+ 1 #本身原子总数

df_X 和 四个参数

sequence_atomic = np.arange(1,stator3_count) #本身原子序列

stator3_distance_origin = df_rotor2_ad.iloc[0][5] + dz_02 #- dz_01 # - dz_01#+ 2*dz_04 # - dz_01#左端与远点的距离

stator3_sequence = stator2_sequence + df_stator2[0].count() #在结构布局里的原子起

始值 等于前面的所有碳纳米管原子总数的和

stator3_first_column = "stator3" #不同部分的第一列不同的赋予不同的 name

df_stator3_ad = adjust_gro_format(df_stator3,sequence_atomic,stator3_distance_origin,stator3_sequence,stator3_first_column)

4、stator4 gro 调整第一列(不同部分分组），第三列和 z 坐标

stator4_count = df_stator4[0].count()+ 1 #本身原子总数

df_X 和 四个参数

sequence_atomic = np.arange(1,stator4_count) #本身原子序列

stator4_distance_origin = df_rotor2_ad.iloc[-1][5] - stator4_length - dz_04 #+ dz_01 #-2*dz_04 #+ dz_01#左端与远点的距离

stator4_sequence = stator3_sequence + df_stator3[0].count() #在结构布局里的原子起始值 等于前面的所有碳纳米管原子总数的和

stator4_first_column = "stator4" #不同部分的第一列不同的赋予不同的 name

df_stator4_ad = adjust_gro_format(df_stator4,sequence_atomic,stator4_distance_origin,stator4_sequence,stator4_first_column)

########################## 电机右边定子 ##########################

5、stator5 gro 调整第一列(不同部分分组），第三列和 z 坐标

stator5_count = df_stator5[0].count()+ 1 #本身原子总数

df_X 和 四个参数

sequence_atomic = np.arange(1,stator5_count) #本身原子序列

stator5_distance_origin = df_rotor3_ad.iloc[0][5] + dz_02 #- dz_01 #左端与远点的距离

stator5_sequence = stator4_sequence + df_stator4[0].count() #在结构布局里的原子起始值 等于前面的所有碳纳米管原子总数的和

stator5_first_column = "stator5" #不同部分的第一列不同的赋予不同的 name

df_stator5_ad = adjust_gro_format(df_stator5,sequence_atomic,stator5_distance_origin,stator5_sequence

,stator5_first_column)

6、stator6 gro 调整第一列(不同部分分组），第三列和 z 坐标

stator6_count = df_stator6[0].count()+ 1 #本身原子总数

df_X 和 四个参数

sequence_atomic = np.arange(1,stator6_count) #本身原子序列

stator6_distance_origin = df_stator4_ad.iloc[-1][5] + 1 + L2 - stator6_length #+ dz_04 #左端与远点的距离

stator6_distance = df_stator4_ad.iloc[-1][5] + 1 + L2 - stator6_length #+ dz_04 #左端与远点的距离

stator6_distance_origin = df_stator4_ad.iloc[-1][5] + 1 + L2 - stator6_length + dz_04 #左端与原点的距离

stator6_distance_origin = df_rotor3_ad.iloc[-1][5] - stator6_length - dz_04

stator6_sequence = stator5_sequence + df_stator5[0].count() #在结构布局里的原子起始值 等于前面的所有碳纳米管原子总数的和

stator6_first_column = "stator6" #不同部分的第一列不同的赋予不同的 name

df_stator6_ad = adjust_gro_format(df_stator6,sequence_atomic,stator6_distance_origin,stator6_sequence,stator6_first_column)

合并后的碳纳米管模型

#合并后的碳纳米管模型

models_ad = [df_motor_ad,df_rotor1_ad,df_rotor2_ad,df_rotor3_ad,

df_stator1_ad,df_stator2_ad,df_stator3_ad,df_stator4_ad,df_stator5_ad,df_stator6_ad

]

根据需要氢化每个模型的末端 ---for 循环--调用函数 find_z_add_hdz

```
#根据需要氢化每个模型的末端
for i,e in enumerate(models_ad):
    if i == 0:
        direction = "R"
        df_motor_H = find_z_add_hdz(models_ad[i],direction,hdz)
#         df_motor_H
    elif i == 1 :
        direction = "LR"
        df_rotor1_H = find_z_add_hdz(models_ad[i],direction,hdz)
    elif i == 2:
        direction = "LR"
        df_rotor2_H = find_z_add_hdz(models_ad[i],direction,hdz)
    elif i == 3:
        direction = "LR"
        df_rotor3_H = find_z_add_hdz(models_ad[i],direction,hdz)
    elif i == 4:
        direction = "L"
        df_stator1_H = find_z_add_hdz(models_ad[i],direction,hdz)
    elif i == 5:
        direction = "R"
        df_stator2_H = find_z_add_hdz(models_ad[i],direction,hdz)
    elif i == 6:
        direction = "L"
        df_stator3_H = find_z_add_hdz(models_ad[i],direction,hdz)
    elif i == 7:
        direction = "R"
        df_stator4_H = find_z_add_hdz(models_ad[i],direction,hdz)
```

```
    elif i == 8:
        direction = "L"
        df_stator5_H = find_z_add_hdz(models_ad[i],direction,hdz)
    elif i == 9:
        direction = "R"
        df_stator6_H = find_z_add_hdz(models_ad[i],direction,hdz)

### 氢化后的模型存储在 models_H 列表

#氢化后的模型存储在 models_H 列表
models_H = [df_motor_H,df_rotor1_H,df_rotor2_H,df_rotor3_H,
            df_stator1_H,df_stator2_H,df_stator3_H,df_stator4_H,df_stator5_H,df_stator6_H]

# 合并

## 合并碳

#合并
# 合并碳
df_ad                                                                                    =
df_motor_ad.append(df_rotor1_ad).append(df_rotor2_ad).append(df_rotor3_ad).\
        append(df_stator1_ad).append(df_stator2_ad).append(df_stator3_ad).\
        append(df_stator4_ad).append(df_stator5_ad).append(df_stator6_ad)
df_ad = df_ad.reset_index(drop=True)



## 合并氢

#合并氢
```

```
df_ad_H = df_motor_H.append(df_rotor1_H).append(df_rotor2_H).append(df_rotor3_H).\
        append(df_stator1_H).append(df_stator2_H).append(df_stator3_H).\
        append(df_stator4_H).append(df_stator5_H).append(df_stator6_H)
df_ad_H = df_ad_H.reset_index(drop = True)

## 调整氢的结构

#调整氢的 gro 格式
# 调整氢的第一列和第三列
for i in range(0,df_ad_H.shape[0]):
#     df_ad_H[0][i] = str(df_ad.shape[0] + i + 1) + "H" + df_ad[0][i][-2:]  # HC 第一列同名不同序列号
    df_ad_H[2][i] = str(df_ad.shape[0] + i + 1)
# 修改第二列 df_ad_h[1]
df_ad_H[1] = df_ad_H[1].replace(to_replace=r'C', value='H', regex=True)

## 结构布局调整完成后合并碳和氢

#结构布局调整完成后合并碳和氢
df_M = df_ad.append(df_ad_H)
df_M = df_M.reset_index(drop = True)

## 输出完整 M 文件 gro

#输出完整 M 文件 gro
os.chdir(path)
# filename_body =  infile_motor.split("_")[-5].split("/")[-1]+infile_motor.split("_")[-3] + "_" + \
# infile_rotor1.split("_")[-5].split("/")[-1]+infile_rotor1.split("_")[-3] + "_" + \
# infile_rotor2.split("_")[-5].split("/")[-1]+infile_rotor2.split("_")[-3]
```

```
# #输出的文件名
outfile_gro = filename_head  + filename_tail
if os.path.exists(filename_head):
# #     os.removedirs(path)
#     os.remove(path)  #删除文件
# #     os.removedirs(path)  #删除空文件夹
#     shutil.rmtree(path)  #递归删除文件夹
#     os.chdir(path)
#     os.mkdir(filename_head)
    pass
else:
    os.mkdir(filename_head)

# [文件目录]  (catalog_2)

os.chdir(path + "\\"+filename_head)
catalog_2 = os.getcwd()

# C、格式化为 VMD 格式函数
dataframe_output_gro(df_M,outfile_gro)




# RS_VDW

models_shape = [len(models_ad[i]) for i in range(len(models_ad))]
```

```
models_shape


start_end = []
for i in range(len(models_shape)):
    if i is 0:
        start_end.append((int(df_M[:1][2]),int(df_M[models_shape[0]-
1:models_shape[0]][2])))
    else:
        start_end.append((start_end[-1][-1]+1,int(df_M[sum(models_shape[:i+1])-
1:sum(models_shape[:i+1])][2])))



models_SE                                                                    =
DataFrame(start_end,columns=["start","end"],index=["motor","rotor1","rotor2","rotor3",
                                                    "stator1","stator1","stator2","stator2",
                                                    "stator3","stator3"])


models_SE[models_SE.index == "motor"]



shutil.copyfile("F:/_python/20190517/M4S1414-88-88-88-88/rs_1.c","rs_1.c")    #  oldfile
和 newfile 都只能是文件
shutil.copyfile("F:/_python/20190517/M4S1414-88-88-88-88/rs_2.c","rs_2.c")    #  oldfile
和 newfile 都只能是文件
shutil.copyfile("F:/_python/20190517/M4S1414-88-88-88-88/rs_3.c","rs_3.c")    #  oldfile
和 newfile 都只能是文件


shutil.copyfile("F:/_python/20190517/M4S1414-88-88-88-88/mr1.c","mr1.c")    #  oldfile
和 newfile 都只能是文件
shutil.copyfile("F:/_python/20190517/M4S1414-88-88-88-88/r1r2.c","r1r2.c")    #  oldfile
和 newfile 都只能是文件
shutil.copyfile("F:/_python/20190517/M4S1414-88-88-88-88/r2r3.c","r2r3.c")    #  oldfile
```

和 newfile 都只能是文件

shutil.copyfile("F:/_python/20190517/M4S1414-88-88-88-88/xdrfile.h","xdrfile.h") # oldfile 和 newfile 都只能是文件

```
import sys
# sys.setdefaultencoding('utf8')
sys.getdefaultencoding()
# sys.setdefaultencoding('Cp1252')

with open("rs_1.c","r+",encoding='Cp1252') as f:
    row_1 = f.readlines()
with open("rs_2.c","r+",encoding='Cp1252') as f:
    row_2 = f.readlines()
with open("rs_3.c","r+",encoding='Cp1252') as f:
    row_3 = f.readlines()



with open("mr1.c","r+",encoding='Cp1252') as f:
    row_4 = f.readlines()
with open("r1r2.c","r+",encoding='Cp1252') as f:
    row_5 = f.readlines()
with open("r2r3.c","r+",encoding='Cp1252') as f:
    row_6 = f.readlines()

## rotor1 and stator1

row_1.pop(28)
row_1.insert(28,"    for  (natom=%s;natom<=%s;natom++)            // 转 子 原 子  ID \n"%(start_end[1][0],start_end[1][1]))
```

row_1.pop(32)

row_1.insert(32,"\t for (natom_1=%s;natom_1<=%s;natom_1++) //定子原子 ID \n"%(start_end[4][0],start_end[5][1]))

row_1[28],row_1[32]

rotor2 and stator2

row_2.pop(28)

row_2.insert(28," for (natom=%s;natom<=%s;natom++) // 转 子 原 子 ID \n"%(start_end[2][0],start_end[2][1]))

row_2.pop(32)

row_2.insert(32,"\t for (natom_1=%s;natom_1<=%s;natom_1++) //定子原子 ID \n"%(start_end[6][0],start_end[7][1]))

row_2[28],row_2[32]

rotor3 and stator3

row_3.pop(28)

row_3.insert(28," for (natom=%s;natom<=%s;natom++) // 转 子 原 子 ID \n"%(start_end[3][0],start_end[3][1]))

row_3.pop(32)

row_3.insert(32,"\t for (natom_1=%s;natom_1<=%s;natom_1++) //定子原子 ID \n"%(start_end[8][0],start_end[9][1]))

row_3[28],row_3[32]

motor and rotor1

start_end

row_4.pop(28)

row_4.insert(28," for (natom=%s;natom<=%s;natom++) // 电 机 ID

```
\n"%(start_end[0][0],start_end[0][1]))

row_4.pop(32)

row_4.insert(32,"\t  for (natom_1=%s;natom_1<=%s;natom_1++)     //转子 1 原子 ID
\n"%(start_end[1][0],start_end[1][1]))

row_4[28],row_4[32]


## rotor1 and rotor2


row_5.pop(28)

row_5.insert(28,"   for (natom=%s;natom<=%s;natom++)          // 转 子  1  原 子  ID
\n"%(start_end[1][0],start_end[1][1]))

row_5.pop(32)

row_5.insert(32,"\t  for (natom_1=%s;natom_1<=%s;natom_1++)     //转子 2 原子 ID
\n"%(start_end[2][0],start_end[2][1]))

row_5[28],row_5[32]


## rotor2 and rotor3


row_6.pop(28)

row_6.insert(28,"   for (natom=%s;natom<=%s;natom++)          // 转 子  2  原 子  ID
\n"%(start_end[2][0],start_end[2][1]))

row_6.pop(32)

row_6.insert(32,"\t  for (natom_1=%s;natom_1<=%s;natom_1++)     //转子 3 原子 ID
\n"%(start_end[3][0],start_end[3][1]))

row_6[28],row_6[32]




with open("rs_1.c","w") as wf:

    for e in row_1:

        wf.write(e)
```

```
with open("rs_2.c","w") as wf:
    for e in row_2:
        wf.write(e)

with open("rs_3.c","w") as wf:
    for e in row_3:
        wf.write(e)



with open("mr1.c","w") as wf:
    for e in row_4:
        wf.write(e)

with open("r1r2.c","w") as wf:
    for e in row_5:
        wf.write(e)

with open("r2r3.c","w") as wf:
    for e in row_6:
        wf.write(e)

row_1[28:33]

row_4[28:33]

row_5[28:33]

row_3[28:33]
```

四、data 文件

```
models = [df_motor_ad,df_rotor1_ad,df_rotor2_ad,df_rotor3_ad,

df_stator1_ad,df_stator2_ad,df_stator3_ad,df_stator4_ad,df_stator5_ad,df_stator6_ad,
        df_motor_H,df_rotor1_H,df_rotor2_H,df_rotor3_H,
        df_stator1_H,df_stator2_H,df_stator3_H,df_stator4_H,df_stator5_H,df_stator6_H]


data_models = pd.DataFrame(columns = [0,1,2,3,4])
for i,e in enumerate(models):
    e = e.drop(columns = [0])  #删除第一列  #保留第二，三，xyz 列
    e[2] = i+1
    e.columns = [0,1,2,3,4]
    data_models = data_models.append(e)
#     if i == 19:
#         print(e)
#         break


data_models[0] = np.arange(1,data_models.shape[0]+1)


data_models[0] = data_models[0].map(lambda x: '%8d' % x)


data_models[1] = data_models[1].map(lambda x: '%6s' % x)


data_models[2] = data_models[2].map(lambda x: '%9.3f' % float(x))
data_models[3] = data_models[3].map(lambda x: '%9.3f' % float(x))
```

```
data_models[4] = data_models[4].map(lambda x: "%9.3f" % float(x))
data_models.columns = ["sequence","type","x","y","z"]

data_models.x = data_models.x.astype(float)
# data_models[3] = data_models[3].astype(float)
data_models.y = data_models.y.astype(float)
data_models.z = data_models.z.astype(float)

data_models = data_models.eval("""
                        x =  x * 10
                        y =  y * 10
                        z =  z * 10
                         """)
data_models.columns = [0,1,2,3,4]

data_models[2] = data_models[2].map(lambda x: "%9.3f" % float(x))
data_models[3] = data_models[3].map(lambda x: "%9.3f" % float(x))
data_models[4] = data_models[4].map(lambda x: "%9.3f" % float(x))

atoms = [ data_models.iloc[i][0] + data_models.iloc[i][1] + \
        data_models.iloc[i][2] + data_models.iloc[i][3] + \
        data_models.iloc[i][4] +"\n"
        for i in np.arange(0,data_models.shape[0])]

############################################################    data 文件
#########################################################################
##


####  mass  ############
box_space = 50
```

```
relative_mass = ""
C_mass = "12.01070"
H_mass = "1.00794"
for i in range(0,total_type):
    if i < int(total_type/2):
        temp = "%3s%9s\n_"%(i+1,C_mass)
    elif i >=  int(total_type/2):
        temp = "%3s%9s\n_"%(i+1,H_mass)
    relative_mass= relative_mass + temp
outfile_data = "data." + filename_head
#定义盒子大小
raw_xyz_range = [[-box_space,box_space ],[-box_space,box_space],
                 [df_motor_ad.iloc[0][5]*10    -    box_space,df_rotor3_ad.iloc[-1][5]*10    +
box_space]]   #  unit:  埃（0.1 纳米）


# 输出：写入到输出文件（outfile）
#  E、 data 文件的  head   格式函数
with open(outfile_data, 'w') as wf:
    data = label(raw_xyz_range,df_M.shape[0],relative_mass) + atoms
    for row in data:
        wf.write(row)


# 五、in 文件


#####################################                    in    文    件
#####################################################################
###############


# 从原点开始，转子 1--》转子 2--》电机--》转子 3--》转子 4--》定子 1--》定子 2--》。。。--》定子 8
```

```
# lable_1  in 文件的 head
C_atoms = " C "*amount
H_atoms = " H "*amount
lable_1 = [
        "units          metal \n",\
        "dimension      3 \n",\
        "timestep       0.001\n",\
        "atom_style     atomic \n",\
        "boundary       s    s    s \n",\
        "neighbor       2.0    bin \n",\
        "neigh_modify    delay    10 \n",\
        "read_data      "+ outfile_data +"\n",\
        "pair_style     hybrid         lj/cut 10.2 airebo 3.0 #10.2\n",\
        "pair_coeff     *    *         airebo CH.airebo "+ C_atoms + H_atoms + "\n"]


########    G、 L-J pair_coeff 格式化   ##########
# label_2     L-J pair_coeff
# serial_number is start
# amount is end
def pair_coeff(type_type,serial_number,amount):
    temp = []
    j = 0
####################   C-C   #######
    if type_type == "C-C":
        epslon = 0.002840
        sigama = 3.400
        if serial_number < amount - 1:
            pair_coeff_C     =     ["pair_coeff"     +     "%5d"%(serial_number)     +     "%5d*%-
5d"%(serial_number + 1,amount) +  "%10s"%("lj/cut") +\
```

```
                    "%10.6f"%(epslon) +  "%10.3f"%(sigama)+"\n"]
        else:
            pair_coeff_C = ["pair_coeff" + "%5d"%(serial_number) + "%8d"%(amount) + "%13s"%("lj/cut") +\
                    "%10.6f"%(epslon) +  "%10.3f"%(sigama)+"\n"]
        temp= temp + pair_coeff_C
#####################  C-H  #########
    elif type_type == "C-H" :
        epslon = 0.001376
        sigama = 3.025
        if serial_number == 1:
            pair_coeff_C = ["pair_coeff" + "%5d"%(serial_number) + "%5d*%-5d"%(amount + serial_number +1 ,2*amount) +  "%10s"%("lj/cut") +\
                    "%10.6f"%(epslon) +  "%10.3f"%(sigama) + "\n"]
        elif serial_number == 2:
            pair_coeff_A = ["pair_coeff" + "%5d"%(serial_number) + "%8d"%(amount + serial_number - 1) +  "%13s"%("lj/cut") +\
                    "%10.6f"%(epslon) +  "%10.3f"%(sigama) + "\n"]
            pair_coeff_B = ["pair_coeff" + "%5d"%(serial_number) + "%5d*%-5d"%(amount + serial_number + 1,2*amount) +  "%10s"%("lj/cut") +\
                    "%10.6f"%(epslon) +  "%10.3f"%(sigama) + "\n"]
            pair_coeff_C = pair_coeff_A + pair_coeff_B
        elif serial_number > 2 and serial_number < amount -1 :
            pair_coeff_A = ["pair_coeff" + "%5d"%(serial_number) + "%5d*%-5d"%(amount + 1,amount + serial_number -1) +  "%10s"%("lj/cut") +\
                    "%10.6f"%(epslon) +  "%10.3f"%(sigama) + "\n"]
            pair_coeff_B = ["pair_coeff" + "%5d"%(serial_number) + "%5d*%-5d"%(amount + serial_number + 1,2*amount) +  "%10s"%("lj/cut") +\
                    "%10.6f"%(epslon) +  "%10.3f"%(sigama) + "\n"]
            pair_coeff_C = pair_coeff_A + pair_coeff_B
        elif serial_number ==  amount -1:
```

```
            pair_coeff_A = ["pair_coeff" + "%5d"%(serial_number) + "%5d*%-5d"%(amount  +
1,2*amount-2) +  "%10s"%("lj/cut") +\
                    "%10.6f"%(epslon) +  "%10.3f"%(sigama) + "\n"]
            pair_coeff_B  =  ["pair_coeff"  +  "%5d"%(serial_number)  +  "%8d"%(2*amount)  +
"%13s"%("lj/cut") +\
                    "%10.6f"%(epslon) +  "%10.3f"%(sigama) + "\n"]
            pair_coeff_C = pair_coeff_A + pair_coeff_B
        else :
            pair_coeff_C  =  ["pair_coeff"  +  "%5d"%(serial_number)  +  "%5d*%-5d"%(amount  +
1 ,2*amount-1) +  "%10s"%("lj/cut") +\
                    "%10.6f"%(epslon) +  "%10.3f"%(sigama) + "\n"]
        temp= temp + pair_coeff_C
#####################  H-H  ##########
    elif type_type == "H-H" :
        epslon = 0.001500
        sigama = 2.650
        if serial_number < amount - 1:
            pair_coeff_C  =  ["pair_coeff"  +  "%5d"%(serial_number+amount)  +  "%5d*%-
5d"%(serial_number + 1+ amount, 2*amount) +\
                    "%10s"%("lj/cut") + "%10.6f"%(epslon) +  "%10.3f"%(sigama) + "\n"]
        else:
            pair_coeff_C = ["pair_coeff" + "%5d"%(serial_number+amount) + "%8d"%(2*amount)
+\
                    "%13s"%("lj/cut") + "%10.6f"%(epslon) +  "%10.3f"%(sigama) + "\n"]
        temp= temp + pair_coeff_C
    return temp


# G、 L-J pair_coeff 格式化
############################


# C-C
```

```
C_C = []
for i in range(1,amount):
    serial_number =i
    amount =amount
    C_C.append(pair_coeff("C-C",serial_number,amount))
pair_coeff_C_C = C_C
# C-H
C_H = []
for i in range(1,amount+1):
    serial_number =i
    amount =amount
    C_H.append(pair_coeff("C-H",serial_number,amount))
pair_coeff_C_H = C_H
# H-H
H_H = []
for i in range(1,amount):
    serial_number =i
    amount =amount
    H_H.append(pair_coeff("H-H",serial_number,amount))
pair_coeff_H_H = H_H
################################
pair_coeff = pair_coeff_C_C + pair_coeff_C_H + pair_coeff_H_H
temp = []
for i,e in enumerate(pair_coeff):
    for j,row in enumerate(e):
        temp.append(row)

pair_coeff = temp   ############ 嵌套列表转换为字符串类型
lable_2 = pair_coeff
```

```
# label_3  define region five and two carbon atom rings length is 4.91A
# dz 是定义固定碳纳米管的宽度
dz = motor_length/6 *10 #4.910

df_motor_ad[3] = df_motor_ad[3].apply(float).map(lambda x: x )
df_motor_ad[4] = df_motor_ad[4].apply(float).map(lambda x: x )
MC_px = pow((pow(max(df_motor_ad.iloc[:][3]),2)+pow(min(df_motor_ad.iloc[:][3]),2)),1/2)*10
MC_pz = (df_motor_ad.iloc[0][5]) *10
motor_region = ["region        motor_L    block" \
          + "%10.3f"%(-MC_px) + "%10.3f"%(MC_px)\
          + "%10.3f"%(-MC_px) + "%10.3f"%(MC_px) \
          + "%10.3f"%(MC_pz ) + "%10.3f"%(MC_pz + dz)+"\n"]
# H、转子左右两端固定区域函数
rotor1_region = X_region(df_rotor1_ad,"rotor1",dz)
rotor2_region = X_region(df_rotor2_ad,"rotor2",dz)
rotor3_region = X_region(df_rotor3_ad,"rotor3",dz)


lable_3 =  motor_region + rotor1_region + rotor2_region + rotor3_region

#lable_4    define groups
group_C = ["motor","rotor1","rotor2","rotor3",\
          "stator1","stator2 ","stator3","stator4","stator5","stator6 "]
group_H = ["motor_hydrogen","rotor1_hydrogen","rotor2_hydrogen","rotor3_hydrogen",
          "stator1_hydrogen","stator2_hydrogen","stator3_hydrogen","stator4_hydrogen",\
         "stator5_hydrogen","stator6_hydrogen"]
#I、根据原子类型分组
```

```
define_groups_C = group_type("C",group_C,amount)
define_groups_H = group_type("H",group_H,amount)
type_groups = define_groups_C+define_groups_H
lable_4 = type_groups

#label_5   region groups
region_groups =  ["group       motor_Left          region        motor_L\n",\
                 "group       rotor1_Left          region        rotor1_L\n",\
                 "#group       rotor1_Right          region        rotor1_R\n",\
                 "group       rotor2_Left          region        rotor2_L\n",\
                 "#group       rotor2_Right          region        rotor2_R\n",\
                 "group       rotor3_Left          region        rotor3_L\n",\
                 "#group       rotor3_Right          region        rotor3_R\n"]

union_groups =   ["group       M_H                  union         motor     motor_hydrogen\n",\
                 "group       all_subtract_M_H       subtract       all       M_H\n",\
                 "#group       all_subtract_M        subtract       all       Motor\n"\

                 "group       stators1             union         stator1 stator2 \n",\
                 "group       stators2             union         stator3 stator4 \n",\
                 "group       stators3             union         stator5 stator6 \n",\

                 "group        except_rotor1          union          motor    stators1 rotor2 stator3
\n",\
                 "group        except_rotor2          union          rotor1   stator2    stators2 rotor3
stator5 \n",\
                 "group       except_rotor3         union         rotor2 stator4 stators3 \n",\
                ]
fix_spring = [ "fix       spring_ML           motor_Left         spring/self 1000    xyz\n",\
              "dump     dump_minimize       all            xtc       200     dump_minimize.xtc\n",\
```

```
"minimize 1.0e-12 1.0e-12 10000 100000\n",\
"#minimize 1.0e-4 1.0e-6 10000 100000\n",\
"#min_modify dmax 0.1\n",\
"undump   dump_minimize\n",\
"fix      spring_ML          motor_Left        spring/self 1000    xyz\n",\
"fix      spring_R1L         rotor1_Left       spring/self 1000    xyz\n",\
"#fix     spring_R1R         rotor1_Right      spring/self 1000    xyz\n",\
"fix      spring_R2L         rotor2_Left       spring/self 1000    xyz\n",\
"#fix     spring_R2R         rotor2_Right      spring/self 1000    xyz\n",\

"fix      spring_R3L         rotor3_Left       spring/self 1000    xyz\n",\
"#fix     spring_R3R         rotor3_Right      spring/self 1000    xyz\n",\

"fix      spring_stator1     stator1           spring/self 1000    xyz\n",\
"fix      spring_stator2     stator2           spring/self 1000    xyz\n",\
"fix      spring_stator3     stator3           spring/self 1000    xyz\n",\
"fix      spring_stator4     stator4           spring/self 1000    xyz\n",\
"fix      spring_stator5     stator5           spring/self 1000    xyz\n",\
"fix      spring_stator6     stator6           spring/self 1000    xyz\n",\

"fix      NVE                all               nve \n",\
"fix      NVE_TEMP           all               temp/rescale    200           300 300  1.0
1.0 \n",\
"#thermo_style custom step  temp    etotal \n",\
"#thermo 200 \n",\
"dump     dump_NveTemp       all           xtc      200    dump_NveTemp.xtc\n",\
"run      200000             # 02million\n",\
"#run     100000             # 01million\n",\
"#run     10000              # 10thousand\n",\
```

```
            "#run      1000                #1hundred\n",\
            "unfix    NVE\n",\
            "unfix    NVE_TEMP\n",\
            "undump   dump_NveTemp\n"]
unfix = [
        "unfix spring_ML\n",\
        "unfix spring_R1L\n","#unfix spring_R1R\n",\
        "unfix spring_R2L\n","#unfix spring_R2R\n",\
        "unfix spring_R3L\n","#unfix spring_R3R\n",\
        ]
lable_5 = region_groups + union_groups + fix_spring + unfix


# second half
# label_6
fix = ["fix   spring_ML_z motor_Left       spring/self     1000           z\n",\
      "#fix   M_H_temp    M_H               temp/rescale   200            300 300  1.0 1.0\n",\
      "#fix   NVT         all_subtract_M_H   nvt           temp           300. 300. 0.1\n",\
      "fix   NVT         all               nvt           temp           300. 300. 0.1  #tchain 1 #drag
0.5\n",\
      "fix   rotate      M_H               move rotate   0.0 0.0 0.0     0.0 0.0 1.0     5 \n"]
compute = ["#compute      cc1          all     chunk/atom       type\n",\
          "#compute      torque       all     torque/chunk     cc1\n",\
          "#fix          torque_1     all     ave/time         1 200 200      c_torque[*]      file  %s
mode vector\n"\
          %(filename_head + "_torque.vector"),\
          "compute      cc2          all     chunk/atom       type\n",\
          "compute      mass_center  all     com/chunk        cc2\n",\
          "fix          center_2     all     ave/time         1 200 200      c_mass_center[*]  file   %s
mode vector\n"\
          %(filename_head + "_mcenter.vector"),\
```

```
"compute cc3 all chunk/atom type\n",\
"compute omiga all omega/chunk cc3\n",\
"fix omiga_3 all ave/time 1 200 200 c_omiga[*] file %s mode vector\n"\
%(filename_head + "_omiga.vector"),\
"#compute crs1 rotor1 group/group stators1\n",\
"#compute crs2 rotor2 group/group stators2\n",\
"#compute crs3 rotor3 group/group stators3\n",\
"#fix crs1_scalar stators1 ave/time 1 200 200 c_crs1 file crs1.scalar\n",\
"#fix crs2_scalar stators2 ave/time 1 200 200 c_crs2 file crs2.scalar\n",\
"#fix crs3_scalar stators3 ave/time 1 200 200 c_crs3 file crs3.scalar\n",\


"#compute cmr1 rotor1 group/group motor\n",\
"#compute cr1r2 rotor2 group/group rotor1\n",\
"#compute cr2r3 rotor3 group/group rotor2\n",\
"#fix cmr1_scalar rotor1 ave/time 1 200 200 c_cmr1 file cmr1.scalar\n",\
"#fix cr1r2_scalar rotor2 ave/time 1 200 200 c_cr1r2 file cr1r2.scalar\n",\
"#fix cr2r3_scalar rotor3 ave/time 1 200 200 c_cr2r3 file cr2r3.scalar\n",\

"#compute cer1 rotor1 group/group except_rotor1\n",\
"#compute cer2 rotor2 group/group except_rotor2\n",\
"#compute cer3 rotor3 group/group except_rotor3\n",\
"#fix cer1_scalar except_rotor1 ave/time 1 200 200 c_cer1 file cer1.scalar\n",\
"#fix cer2_scalar except_rotor2 ave/time 1 200 200 c_cer2 file cer2.scalar\n",\
"#fix cer3_scalar except_rotor3 ave/time 1 200 200 c_cer3 file cer3.scalar\n",\

"#thermo_style custom step temp etotal \n",\
"#thermo 200\n"]
```

```
dump =  ["#dump        1          all             custom   200    %s  type x y z fx fy fz\n"\
        %(filename_head + ".lammpstrj"),\
        "#dump        2          all             xtc      4000   dump_per4000_2million.xtc\n",\
        "#dump        3          all             xtc      1000   dump_per1000_2million.xtc\n",\
        "#dump        4          all             xtc      500   dump_per500_2million.xtc\n",\
        "dump        5          all             xtc      200    dump_per200_2million.xtc \n",\
        "#dump        6          all             xyz      200    dump_%s.xyz\n" % (filename_head),\
        "\n#dump_modify 1          element        " + C_atoms + H_atoms + "sort id\n",\
        "\n#restart     50000"+ "%6s"%(filename_head)+".restart",\
        "\n#run         100000\n\n",\
        "\n#restart     100000"+ "%6s"%(filename_head)+".restart",\
        "\n#run         200000\n\n",\
        "run        20000200   #20million\n",\

        "#run        100000     #01million\n",\
        "#run        10000      #10thousand\n",\
        "#run         1000\n",\
        "uncompute   cc2\n",\
        "uncompute   mass_center\n",\
        "uncompute   cc3\n",\
        "uncompute   omiga\n",\
        "undump       5\n"
        ]
lable_6 = fix + compute + dump

# 输出 in 文件：写入到输出文件（outfile）
outfile_in = "in." + filename_head
with open(outfile_in, 'w') as wf:
    lables =  ["#lable_1\n"] + lable_1 + ["#lable_2\n"] +  lable_2 + ["#lable_3\n"] +  lable_3
```

```
+ \
          ["#lable_4\n"] +  lable_4 + ["#lable_5\n"] + lable_5 + ["#lable_6\n"] + lable_6
    data = lables
    for row in data:
        wf.write(row)

# 六、lsf 文件

# bsub_lsf 文件
outfile_lsf = "lsf."+ filename_head
NP =48
with open(outfile_lsf, 'w') as wf:
    bsub_lsf =  ["#!/bin/sh\n",
                "APP_NAME=intelY_mid\n",
                "NP={} #24".format(NP) + "\n",
                "NP_PER_NODE=12\n",
                'RUN="RAW"\n',
                "source /home-yw/env/intel-2016.sh\n",
                "mpirun -np $NP ./lmp_mpi -i  " + outfile_in + "\n"]
    data = bsub_lsf
    for row in data:
        wf.write(row)

# 七、document 文件 模型设置-路径

# document  about models set
with open("document"+filename_head+".txt","w") as wf:
    data = ["filename_head=" + filename_head,"\n",
            "\n",
```

```
            "catalog_0 =" + catalog_0,"\n",
            "catalog_1 =" +  catalog_1,"\n",
            "catalog_2 =" +  catalog_2,"\n",
            "\n",
            "curfilerename_temp=" + curfilerename_temp,"\n",
            "stators_temp=" + stators_temp,"\n",
            "\n",
            "infile_motor = " + str(infile_motor),"\n",
            "infile_rotor1 = " + str(infile_rotor1),"\n",
            "infile_rotor2 = " + str(infile_rotor2),"\n",
            "infile_rotor3 = " + str(infile_rotor3),"\n",
            "infile_stator1 =" + str(infile_stator1),"\n",
            "infile_stator3 =" + str(infile_stator3),"\n",
            "infile_stator5 =" + str(infile_stator5),"\n",
            "models_SE=\n" +  str(models_SE),"\n",
            "\n",
            "hdz=" + str(hdz),"\n",
            "L2 = " + str(L2),"\n",
            "stator6_distance_origin=" + str(stator6_distance_origin),"\n",
            "original set stator6_distance=" + str(stator6_distance),"\n",
            "box_space=" + str(box_space),"\n",
            "fix xy MC_px=" + str(MC_px),"\n",
            "MC_pz=" + str(MC_pz),"\n",
            "fix width dz=" + str(dz),"\n",
            "catalog_2=" + "\n" + catalog_2,"\n"]
    for row in data:
        wf.write(row)

# 八、rm_cp_bsub 文件
```

```
#  rm_cp_bsub
with open("rm_cp_bsub"+now_time,"w") as wf:
    data = [
            'rm -rf  ./lmp_mpi ./CH.airebo',"\n",
            'echo "##  rm -rf ./lmp_mpi ./CH.airebo  ##"',"\n",
            'cp       /home-yw/users/nsyw236_XZ/software/lammps_2018/lammps/lammps-12Dec18/src/lmp_mpi  ./',"\n",
            'cp       /home-yw/users/nsyw236_XZ/software/lammps_2018/lammps/lammps-12Dec18/potentials/CH.airebo ./',"\n",
            '#echo              "##              cp              /home-yw/users/nsyw236_XZ/software/lammps_2018/lammps/lammps-12Dec18/src/lmp_mpi  ./  ##"',"\n",
            '#echo              "##              cp              /home-yw/users/nsyw236_XZ/software/lammps_2018/lammps/lammps-12Dec18/potentials/CH.airebo ./  ##"',"\n",
            '#echo "##  ls -ahl ./  ##"',"\n",
            'dos2unix ./lsf.*',"\n",
            "chmod +x ./lsf.*","\n",
            'bsub   lsf.*',"\n",
            'bjobs >> jobs{}'.format(now_time),"\n",
            'bjobs | tail {} >> document*'.format(-int(NP/12)),"\n",
            'pwd >> document*',"\n",
            'ls -ahl ./',"\n",
            'tail -5  document*',"\n"
            ]
    for row in data:
        wf.write(row)

os.chdir(catalog_0)
```

```
x = 1.414

y = 9.379

two = 2

(x**two+y**two)**0.5/10


os.getcwd()


# 所有变量
```

executed in 27ms, finished 13:47:18 2019-06-25

```
Using matplotlib backend: Qt5Agg

Populating the interactive namespace from numpy and matplotlib

C:\ProgramData\Anaconda3\lib\site-packages\IPython\core\magics\pylab.py:160:
UserWarning: pylab import has clobbered these variables: ['e', 'f', 'fix']

`%matplotlib` prevents importing * from pylab and numpy

  "\n`%matplotlib` prevents importing * from pylab and numpy"

%whos
```