QM/MM笔记

zjxu@simm.ac.cn

2018.12

配置：Gaussian16 B.01，GaussView 5.0.9

目的：想得到如下的gjf用于作业提交：

H-H-0.271900 0 2.368229 -15.311647 -2.403108 L

N-N--0.415700 0 3.250229 -12.246647 -0.770108 L

C-CT--0.001400 0 3.343229 -10.849647 -0.344108 L H-HC-0.03 1260

C-C-0.597300 0 1.980229 -10.151647 -0.406108 L

O-O--0.567900 0 1.537229 -9.619647 0.609892 L

C-CT--0.015200 0 4.415229 -10.100647 -1.163108 H

C-CA--0.234100 0 4.728229 -6.282647 -1.432108 H

C-C-0.322600 0 5.001229 -5.929647 -0.096108 H

O-OH--0.557900 0 5.185229 -4.622647 0.250892 H

H-HO-0.399200 0 5.166229 -4.034647 -0.489108 H

H-H-0.271900 0 3.157229 -12.415647 -1.765108 L

VDW s6 2.0000 0.2500

VDW br 2.22 0.320

VDW Zn 1.10 0.0125

HrmStr1 n3 s6 353.8 1.632

HrmBnd1 n3 s6 c 59.1 99.82

直接从PDB复合物读入gaussianView，高层小分子的原子类型和电荷信息（上面蓝色部分）缺失，以及小分子的力场参数（上面黄色部分），主要目的是为了得到这些信息。

作业准备工作：

小分子的质子化状态可以通过ligprep得到。

A、小分子(准备原子类型和电荷)

从复合物的gjf中删除大分子（Atom List Editor-选择高层，然后Rows-invert selection-然后删除）。还可以将复合物存为pdb，然后在PyMOL中先设置set retain\_order, 1，之后读入复合物，删除polymer，存为pdb，再用gaussianView打开小分子，存为gjf。

1. 存出小分子的PDB格式：

grep '^HETATM' 3b2i.pdb | grep PLM | awk '$6==150' > 3b2i\_ligand.pdb

将PLM从smiles格式转为mol2格式：

从PDB数据库找到PLM的smiles格式：CCCCCCCCCCCCCCCC(O)=O，然后在chemdraw里转为二维格式：Edit – paste special – SMILES，存为cdx格式，将cdx格式用chem3D打开，存为mol2格式：3b2i\_ligand.mol2。

1. 获得小分子的质子化状态：

01\_ligprep.sh -l 3b2i\_ligand.mol2 -o 3b2i\_ligprep.mol2

在Win系统的替代方案：将3b2i\_ligand.mol2读入maestro，然后Tasks – Ligand Preparation – Generate possible states at target pH（改为结晶实验的pH值+/-0.0）

1. 根据\*\_ligprep.mol2中的质子化状态（重原子的坐标发生了一些变化），在gaussianView中对\*\_ligand.pdb中加H，存为\*\_ligand-gjf.pdb，同时存为\*\_ligand-gjf.gjf，头文件改为：

%chk=3b2i\_ligand

%mem=2000Mb

%nproc=4

# opt hf/6-31g(d) geom=connectivity pop=mk iop(6/33=2,6/42=6, 6/50=1)

然后g16 \*\_ligand-gjf.gjf。查看log文件，看最后有无异常。

1. 提取小分子的电荷：

antechamber -i ../\*log -fi gout -o \*\_lig.prep -fo prepi -c resp –nc 0

nc后面跟小分子的formal charge

antechamber生成的以ANTECHAMBER开头的文件中的原子顺序与gaussian输入中的gjf中的原子顺序一致，而生成的NEWPDB.PDB文件和.prep文件中的原子顺序与gjf文件中原子顺序不一致，另外antechamber输出的原子名称与gjf中的原子名称不完全一致，尤其是H。但antechamber自身输出的原子名称一致。（因此现在很明确，将NEWPDB.PDB中的RESP电荷附加到以ANTECHAMBER\_RESP.AC顺序的原子中；将.prep中的原子类型附加到以ANTECHAMBER\_RESP.AC顺序的原子中）。

ANTECHAMBER\_PREP.AC中含有电荷和原子类型信息。ANTECHAMBER\_RESP.AC相当于将ANTECHAMBER\_PREP.AC中的电荷信息归0。

经比较发现ANTECHAMBER\_PREP.AC中有NEWPDB.PDB中的RESP电荷以及.prep中的原子类型，ANTECHAMBER\_PREP.AC中的原子顺序又和gjf中的一致，因此直接将它的电荷信息和原子类型输出即可。

awk '$1=="ATOM" {print $3, toupper($10) "-" $9}' ANTECHAMBER\_PREP.AC

原子名称 原子类型-电荷

N1 N3--0.297108

O1 O--0.703333

C1 C3--0.031802

1. 提取小分子的力场信息：

parmchk -i \*\_lig.prep -f prepi -o ligand.frcmod -a y

生成的ligand.frcmod文件中有力场参数信息（可用于分子力学/动力学模拟）。

\*\_com.gjf的VDW后加上frcmod文件中NONBON里的信息，Hrmstr1后加上BOND里的信息，HrmBnd1后加上ANGLE里的信息。都增加到复合物的gjf文件后面。

awk '/NONBON/, /NR/' \*mod | awk '{print "VDW", $1, $2, $3}'

awk '/^BOND/, /ANGLE/' \*mod | awk '{print "HrmStr1", $1, $2, $3, $4}' | tr '-' ' '

awk '/^ANGLE/, /DIHE/' \*mod | awk '{print "HrmBnd1", $1, $2, $3, $4, $5}' | tr '-' ' '

小分子的立场信息输入完之后需要加上一两行回车，否则报错。

G09D 中的关键词是pop=mk iop(6/50=1)；文件的末尾（一般在坐标后面空一行写上 xxx.gesp）。

适用于G09和G16的小分子gjf文件：

# Opt HF/6-31g\* Pop=MK IOp(6/50=1)

SIN

-1 1

C1 13.529 47.374 -0.405

O1 12.629 47.356 -1.278

O2 14.879 47.872 -0.578

C2 13.079 46.708 0.910

C3 14.191 45.931 1.666

C4 13.679 45.298 2.936

O3 13.739 45.940 3.995

H11 12.350 46.051 0.719

H12 12.740 47.407 1.539

H13 14.943 46.553 1.885

H14 14.531 45.189 1.089

N 12.810 44.301 2.755

H 13.554 43.738 2.367

C 12.041 43.758 3.871

H 12.147 44.395 4.749

H18 12.374 42.841 4.091

H19 11.075 43.710 3.618

antechamber-ini.esp

antechamber.esp

//gjf结束

antechamber -i antechamber.esp -fi gesp -o sin.prepin -fo prepi -c resp -s 2 -rn SIN -at amber -nc -1

也是生成prepi文件。

备注（根据对1hk4配体的检验来看，加不加nc效果好像没区别：

[zjxu@localhost 1hk4]$ pwd

/mnt/sde/zjxu/wwPDB\_Download/HB\_201801/ligplot\_HB/QM\_MM/1hk4

[zjxu@localhost 1hk4]$ awk '/@<TRIPOS>ATOM/,/@<TRIPOS>BOND/' 1hk4\_ligprep.mol2 | head -n -1 | tail -n +2 | awk '{sum=sum+$NF}END{print "The formal charge for the ligand is: " sum}'

The formal charge for the ligand is: -1

[zjxu@localhost resp]$ pwd

/mnt/sde/zjxu/wwPDB\_Download/HB\_201801/ligplot\_HB/QM\_MM/1hk4/resp

[zjxu@localhost resp]$ head ANTECHAMBER\_PREP.AC

CHARGE -1.00 ( -1 )

Formula: H10 C15 N1 O4 I4

）

12、

若有重要的水分子，在pymol中提取出来，加氢之后，保存，再和蛋白质及小分子共同存成一个pdb。最后用GW打开，生成优化文件。

小分子中有I的计算有些特别，gjf文件按照模板修改（所有）

14、小分子中所有的原子类型的参数都列在gjf文件的最后，参数在202：

1) cd $AMBERHOME

2) cd dat/leap/parm

3) vi gaff.dat *#小分子力场参数（vDW、键长、键角、二面角等）*

4) vi parm99.dat *#蛋白中的力场参数（vDW、键长、键角、二面角等）*

（15、所有原子的类型和电荷都全了之后，提交作业；跑死了之后查看log文件，会有undefined bond 和angle，记下原子编号，gjf中找出对应的原子类型（注意编号要加8），gaff.dat文件中找出参数，加在gjf的后面（注意无重复），格式见模板

(undefined出错之后，

./../../../script/undefinedListKiller.pl -gi gjf -go log -o outputFile)）

B、蛋白质

1. 删除蛋白质的H、删除alternative conformation，并利用chimera补全缺失的侧链原子：

fix\_protein.sh 1cxq\_polymer.pdb

1. 利用modeller补全缺失的missing residues，并且只优化missing residues，或者比missing residues再长n个氨基酸（对于1cxq，n=1比较合适）。

在Python3下：

python missingResidues.py

将生产的pdb拷贝为1cxq\_fill\_modeller.pdb，然后赋给它A链，并改回原始的PDB编号：

pymol -rkqc ~/sh/pymol/save\_polymer.py -- 1cxq\_fill\_modeller.pdb A 51

1. 再重新执行一次fix\_protein.sh: fix\_protein.sh 1cxq\_fill\_modeller\_clear.pdb
2. 计算蛋白质的质子化状态（pdb2pqr是比H++更好的选择，H++无法处理缺失氨基酸的蛋白质。蛋白质的formal charge必须准确，否则多重度计算错误。

）。

pdb2pqr --ff=amber --ffout=amber --chain --with-ph=7.5 1cxq\_fill\_modeller\_clear\_fix.pdb 1cxq\_polymer\_fix.pqr

pdb2pqr，在QM/MM中，我们设置的力场是amber，因此这里处理的力场以及输出的力场都设置为amber，--chain为保留原始的原子编号（不从1开始，如果不设置，默认从1开始），--with-ph=7.0为结晶状态下的pH值，并自主处理末端氨基酸，即在C末端增加OXT（对于中间缺失的loop，不会增加OXT）。输出为\*.pqr。

将pqr转化成pdb格式：

obabel -ipqr 1cxq\_polymer\_fix.pqr -o pdb -O 1cxq\_polymer\_obabel.pdb

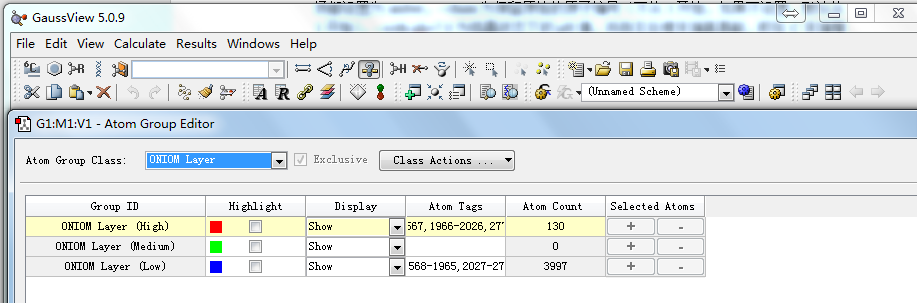
1. 合并蛋白质-配体：

obabel -ipdb 1cxq\_polymer\_obabel.pdb 1cxq\_ligand.pdb -opdb -j -O 1cxq\_com.pdb

1. GV打开PDB文件，默认小分子在高层，蛋白在低层
2. 关于质子化状态，可以在菜单栏的Edit > PDB Residues…里直接编辑，还可以编辑末端AA残基
3. ~~打开Atom list Editor，点（show ONIOM columns），确定高层原子。根据residue信息更方便选择高层原子，高层的H也根据residue信息来选取。~~高低层的连接处会自动显示连接H（）

在Edit – PDB Residues中将不敢兴趣的氨基酸残基都隐藏起来，只show感兴趣的残基，方便进行鼠标操作。

Edit-Atom groups-ONIOM Layer，用鼠标选中要增加到高层的原子，点击 ONIOM Layer （high）行的“+”即可。



1. 先优化结构：Calculate > Gaussian Calculation Setup…
2. Job type: Optimization
3. Method: Multilayer ONIOM Model;
4. High layer: DFT > B3LYP > 6-311G+(d), charge根据具体体系定，保证spin（多重度）为1
5. Low layer: Mechanics > Amber, 其实后面在文本里还要修改(打开文件后，calculate里gaussian calculation setup中的Method里勾选右上角的Multilayer ONIOM Model，Low Layer里的method第二行选择Mechanics，第三行选Amber，这样才能在文本文件中出现原子电荷和原子类型的信息。)
6. Title自定
7. Link: Memory Limit(2400MB); Chk file(specify, 去掉full path); Shared Processors(4)
8. Retain, save gjf, 勾选Write Cartesians
9. 文本打开gjf，按照模板修改头文件（前8行）:

%chk=receptor\_Br\_Et.chk

%mem=8000Mb

%nproc=8

#opt oniom(b3lyp/6-31g\*:amber=hardfirst)

如果标准氨基酸的原子类型和电荷有缺失，去下面的文件里拷贝：

/home/zjxu/prog/amber16/dat/leap/lib/all\_amino94.lib （标准氨基酸）

/home/zjxu/prog/amber16/dat/leap/lib/all\_aminoct94.lib （位于C末端的标准氨基酸）

/home/zjxu/prog/amber16/dat/leap/lib/all\_aminont94.lib （位于N末端的标准氨基酸）

或者如下的三个文件：

/home/zjxu/prog/amber16/dat/leap/lib/all\_amino94ildn.lib

/home/zjxu/prog/amber16/dat/leap/lib/all\_aminoct94ildn.lib

/home/zjxu/prog/amber16/dat/leap/lib/all\_aminont94ildn.lib

主要是断的loop的两端的原子类型和电荷信息有缺失。

查找缺失的原子类型：

[zjxu@localhost 1cxq]$ grep '\-(' 1cxq\_com.gjf

查找缺失的电荷：

[zjxu@localhost 1cxq]$ grep '[^0-9](' 1cxq\_com.gjf

优化SCF无法收敛，可能是因为补的loop结构不大好，用PM6先优化这段loop：

[zjxu@localhost 1cxq]$ awk -f freeze.awk 1cxq\_com-2.gjf > 1cxq\_com-3.gjf，然后修改一下头

Tip：遇到不收敛的情况，先在sybyl中将大分子和小分子整体优化之后，若发现原子类型和电荷缺失，找到缺失部位的原子，在pdb打开文本，一一对应，确定是哪种氨基酸，在pdb里找到相同的氨基酸的原子，在gif中直接复制粘贴到缺失部位即可

用更高精度的基组和方法继续优化：

http://gaussian.com/faq2/

**“Restarting” a Geometry Optimization from a Specific Point: Geom=(AllCheck,Step=n)**

If you want to begin a new geometry optimization from a specific point in a previous optimization, this can be accomplised via the Step option to the Geom keyword. It does not matter if the previous optimization completed, was interrupted or failed.

The following example optimizes the structure at point 20 of the previous optimization:

%OldChk=myfile

%Chk=mynewfile

# method/basis Opt Geom=(AllCheck,Step=20) Guess=Read

You may add any desired options to the Opt keyword. In this example, we use the %OldChk directive to retrieve the desired structure from an existing checkpoint file without modifying that file; a different checkpoint file is used for the new calculation. You can also use the Check option to Geom rather than AllCheck if you prefer (providing the additional required input).

Note that this method does not restart the optimization but rather begins a new one from a specific retrieved geometry.

[zjxu@R820 2piu]$ cat 2piu\_com\_m062x.gjf

%OldChk=2piu\_com.chk

%Chk=2piu\_com\_ m062x.chk

%mem=8000MB

%nprocshared=8

# opt oniom(m062x/6-311+g(d,p):amber=hardfirst) Geom=AllCheck

对于含I的体系：

[zjxu@R820 2piv]$ cat 2piv\_com\_m062x.gjf

%OldChk=2piv\_com.chk

%Chk=2piv\_com\_m062x.chk

%mem=8000Mb

%nproc=8

# opt oniom(m062x/genecp:amber=hardfirst) Geom=AllCheck

C H O N S 0

6-311+g(d,p)

\*\*\*\*

I 0

LANL2DZ

\*\*\*\*

I 0

LANL2DZ

4oh5的m062x还没跑完。

优化完成后处理：

16、算完之后，提取构象：

* 1. $ ./ytliu/gaussian/scripts/gjfGenerator.sh gin \*gjf gout \*log dout . *#本目录下生成gjfs目录，里面有每一轮优化的结构*~/script/gjfGenerator.sh gin \*gjf gout \*log dout .
  2. 将gjfs目录下数目最大的gjf拷到上一层，重命名为\*\_out.gjf
  3. $ ./ytliu/gaussian/scripts/pickupHighLayer.sh –i \*\_out.gjf –o \*\_out\_hlayer.gjf

~/script/pickupHighLayer.sh -i \*\_out.gjf -o \*\_out\_hlayer.gjf

17、将\*hlayer.gjf拷到本地，修改，计算4个能量（复合物、AA、小分子的单点能以及BSSE）

18、GV打开\*hlayer.gjf，将与低层相连的C变成H，按模板编辑gjf

19、文本打开gjf，去掉原子类型和电荷

20、在\*com\_sp.gjf的基础上编辑bsse.gjf，关键词多counterpoise=2，第8行分段，1、2位为整体的charge和多重度；3、4位为第一片段；5、6位为第二片段。一般将ligand作为一个片段，其余作另一片段

21、计算优化过程中两个原子之间的距离：找到原子编号；qmmm目录下，运行optProfileAnalyzer.pl –i atomID1 atomID2 –d gjfs –o <outfile>

22、生成小分子势能面：

1) singlePoint/lig目录下：$ formchk \*chk *#得到fchk文件，下载本地*

2) GV打开，result > surface/contours…> Cube actions > new cube，选择Total density, SCF, use full density matrix, fine

3) cube生成后存一下

4) Surface action: New-mapped surface, ESP, SCF

5) 存成tif格式没有bar，截图存一下

23、一般机器上交作业：

$nohup g09 \*gjf &

24、小机器（8核，可超线程到16核）：

1) 202.127.19.97, 端口：22225, ytliu, 19860830

注意事项：

1. 最好一次操作成功，如果分成两次，注意添加或者替换原子的顺序，会影响原子的编号，导致后面一连串的错误
2. 有二硫键的蛋白，注意pdb文件中要标出（SSBOND）
3. 有的蛋白侧链有两种取向（DS可以看出）以及侧链不全的，可以在spdbv中打开另存一下
4. 中间出现最多的是优化过程中不收敛，问了技术支持后，被告知需要加上所有小分子的原子类型参数，而不仅仅是金属及I。加上后就没问题了。之前做过的尝试有先整体在MM中优化一下再做QM/MM的优化，不太能解决问题
5. GV操作技巧 Alt+左键可以转动一个（水）分子。Alt+Shift+左键可以平动一个分子。
6. G03在某些linux系统下可能会出现下列错误：Erroneous write during file extend. write xxxxx instead of 4096

问题的解决方法是用root用户执行下列命令：

echo 0 > /proc/sys/kernel/randomize\_va\_space

也可以将这一句写入/etc/rc.local文件，每次开机自动运行，就不用再bash里面每次都输入了

Gaussian tips:

1. opt的maxcyc是指结构优化时程序自动调整体系的全部核坐标的次数，调整的目的是找到能量最低的局部极小点
2. scf的maxcyc是指对于一个核坐标已确定了的体系在波恩-奥本海默近似的前提下，解自恰场方程时自洽迭代的次数

.

计算RMSD：

将初始构象的PDB中的坐标替换成优化好的结构的坐标，存成新的PDB，注意TER。从优化好结构的pdb中拷贝坐标到初始构象的pdb中，应严格注意其中的空格多少，pdb的格式非常严格，所以一般从整数的30位拷贝到55位，否则将会出错。

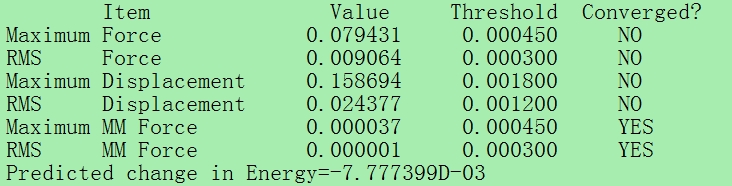
将现在的pdb与初始pdb进行align，Action-align-to molecule，点all-S-organic-sticks，将小分子突出显示，分别在两个分子中选中相互作用的小分子和残基，A-copy to object，在新的object中删去多余的原子，使之与高层结构一致，再运行rmsd\_b.py，命令为：

run D:\rmsd\_b.py (rmsd\_b.py脚本的路径在D盘)

rmsd\_b selection1, selection2

得到RMSD值。

PDB中的小分子若是药，在PDB中的Ligand Chemical Component中，直接点击小分子的名称，进去之后，就有Drug Info: DrugBank的信息。

分析优化程度：在log文件中搜索YES关键字，出现

右侧出现六个YES时表示优化已完成。

当Value里的值与Threshold相差不多时较好。而Predicted change inEnergy在-\*.\*\*\*D-06-09才成功。

计算Single-poing PCM solvation energy时，先计算complex的单点能，再计算两个单体的单点能。算单体的能量时，在输入文件中，将复合物所有原子坐标都加入，但在需要隐藏的相互作用的配体（或受体）的原子类型后加入-Bq，需要注意的是原子电荷和多重度，表示的是需要计算部分的电荷及多重度，而非整体的。算出的能量用complex-lig-recp。

计算真空中的能量时，也可以用这种方法。还可以算出complex、lig及recp（不含相互作用的原子）分别的单点能，再算BSSE，最后用complex-lig-recp-BSSE\_energy得到相互作用能。

计算单点能用mp2的方法算出来的log文件最后得到的能量有两个，一个为HF=……，一个为mp2=……，需要用后面mp2的数值进行计算。MP2方法就是在HF基础上再做的，所以MP2后面的能量。

Pymol中将距离数值的小数位数改为2，可输入 set label\_distance\_digitals, 2

tips：

因缺失一段氨基酸，H++无法识别。

PyMOL>count\_atoms name ca and (r. arg or r. lys)

count\_atoms: 19 atoms （正电荷为19个）

PyMOL>count\_atoms name ca and (r. asp or r. glu)

count\_atoms: 10 atoms （负电荷为10个）

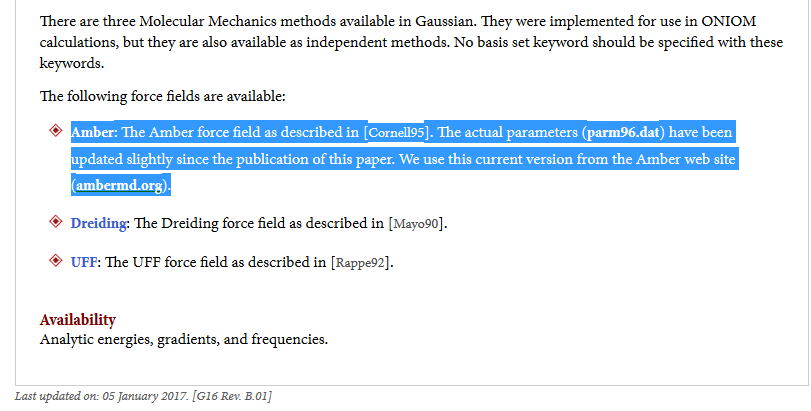
PyMOL>count\_atoms name ca and r. His（肉眼观察，无质子化）

因此整体电荷为+9。低层电荷为+9（所有的原子都在低层），高层电荷为+1.

G16B.01中的Ambe分子力场：

<http://gaussian.com/mm/>

The actual parameters (parm96.dat) have been updated slightly since the publication of this paper. We use this current version from the Amber web site ([ambermd.org](http://ambermd.org)).



朱正诞20190103：

ONIOM的简要流程我之前整理了一个，供参考：

