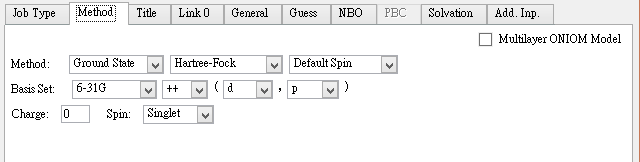
**Ligand建立**

使用GaussView5.0建立ligand的結構，將建立好的結構進行optimization。

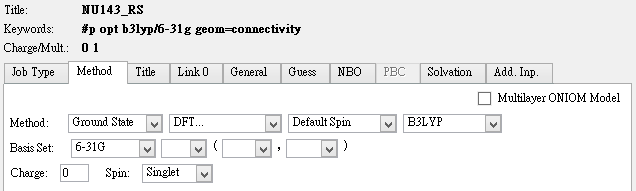
右鍵=>Calculate=>Gaussian Calculation Setup

Job Type: Optimization

Method:



Or

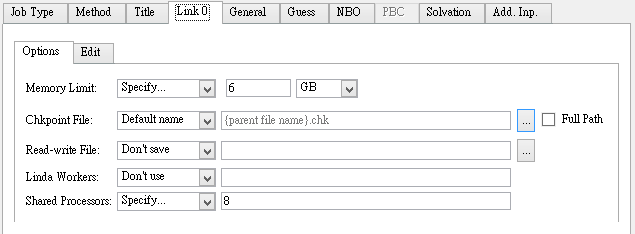


Title: 輸入Job的名稱

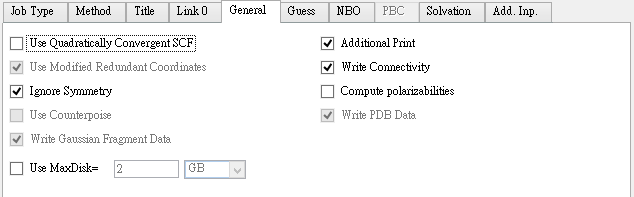
Link 0: Full Path要取消，若勾選gjf檔案會出現路徑，會出錯

Memory Limit: 記憶體大小，若沒設定默認為256MB

Shared Processors: 要使用的CPU數目



General:



Add. Inp: scf=tight (可設可不設)

其他設定不動=>Edit=>Save ligand.gjf檔

將drug.gjf上傳至機房計算=>g09sub ligand，不需輸入副檔名

計算完會產生ligand.log與ligand.chk檔案

將ligand.log從機房複製至電腦中，使用GaussView讀取，檢查是否有錯誤，確認結構為正確的後，轉存為ligand.pdb檔

右鍵=>File=>Save，檔案名稱: ligand.pdb，Save as: PDB File

使用VMD 1.9.2開啟ligand.pdb，在VMD Main=>Extensions=>Tk Console輸入指令更改resname、resid及segname，可方便後續分析

指令

set aaa [atomselect top “all”] => aaa為變數

$aaa set resname AAA

###在pdb檔案因為格式的限制，atom name至多為三個字元

set aaa [atomselect top “all”]

$aaa set resid 1

$aaa get resid => 可看resid

set aaa [atomselect top “all”]

$aaa set segname LIG

將更改好後的檔案存為新的.pdb檔

File=>Save Coordinates，File type: pdb=>Save，檔案名稱: drug.pdb

為resname、resid及segname命名好後，再來需要給atom序號

在該檔案資料夾下開啟命令視窗，輸入perl reorder\_drag\_pdb.pl input > output

Output的pdb檔為最終藥物的PDB檔

reorder\_drag\_pdb.pl

use strict;

($#ARGV != 0)?

die "Please give your pdb filename.\n":

open(INF, "$ARGV[0]") || die "Can't open $ARGV[0].\n";

my $order = 1;

my %num\_class;

while(my $line=<INF>){

if($line =~ /^ATOM/ || $line =~ /^HETATM/){

my @word=split(//, $line);

my $name;

for(my $i=13; $i<=15; $i++){

if($word[$i] ne ' '){

$name.=$word[$i];

}

else{

last;

}

}

if(exists $num\_class{$name}){

$num\_class{$name}++;

}

else{

$num\_class{$name} = 1;

}

my @reorder = split(//, $num\_class{$name});

for(my $i=13; $i<=15; $i++){

if($word[$i] eq ' '){

if($#reorder >=0){

$word[$i] = shift @reorder;

}

}

}

print @word;

}

else{

print $line;

}

}

**Topology與Parameter建立**

到CGenFF建立藥物的Topology與Parameter

網址: <https://cgenff.paramchem.org/>

CGenFF=>Upload molecule，登入後將藥物的pdb檔(drug.pdb)上傳，選擇Include parameters that are already in CGenFF=>Upload File，會產生選Output檔drug.str

drug.str為藥物的Topology與Parameter，上方為Topology，下方為Parameter，右鍵另存新檔後將Topology與Parameter再分開另存，topology的/scrat需更改為藥物名稱，將Topology的CHARGE更改為Gauss計算出來的ligand.log中的Mulliken atomic charges的電荷

※是否勾選Include parameters that are already in CGenFF的差別在於，不打勾會刪掉與parameters重疊的部分

※可能產生的問題: CGenFF產生的電荷為小數點後三位，Gauss計算出來的電荷為小數點後六位，因此若使用Gauss的電荷，整體的電荷總數就可能不會為0或不是整數，當後續在做自由能計算時，NMA在產生charmm的設定檔時就會因為電荷的問題而產生錯誤

**結構能量最小化**

需先建立藥物的psf，在命令視窗輸入vmd –e 02\_addHAndGetPSF\_chainX.txt，會讀取topology與藥物的.pdb檔，寫出.pdb和.psf

02\_addHAndGetPSF\_chainX.txt

package require psfgen[[1]](#footnote-1)

topology top\_all27\_prot\_na.txt

topology top\_all36\_cgenff\_400.txt

topology drug\_top.txt[[2]](#footnote-2)

pdbalias residue HIS HSE

pdbalias residue CSW CYS

pdbalias residue CSO CYS

pdbalias atom ILE CD1 CD[[3]](#footnote-3)

segment X {pdb drug.pdb;first none;last none}[[4]](#footnote-4)

coordpdb drug.pdb X[[5]](#footnote-5)

guesscoord[[6]](#footnote-6)

writepdb drug\_H.pdb

writepsf drug\_H.psf [[7]](#footnote-7)

exit[[8]](#footnote-8)

再將寫出的drug\_H.pdb進行minimize，使結構能量最佳化

01\_mini.namd

structure drug\_H.psf

coordinates drug\_H.pdb

temperature 0

paraTypeCharmm on

parameters par\_all27\_prot\_na.txt

parameters par\_all36\_cgenff\_400.txt

parameters zinc\_finger\_par.txt

parameters drug\_par.txt

outputEnergies 1000

outputTiming 1000

xstFreq 1000

dcdFreq 1000

timestep 1

nonBondedFreq 2

fullElectFrequency 4

stepsPerCycle 20

switching on

switchDist 10

cutoff 12

pairlistdist 14

exclude scaled1-4

1-4scaling 1.0

binaryoutput off

outputname drug\_H\_mini\_out

restartname drug\_H\_mini\_rst

restartfreq 1000

minimize 10000

會得到drug\_H\_mini\_out與drug\_H\_mini\_rst，副檔名含coor,dcd,vel,xsc,xst，後續需要.pdb檔，而.coor檔為整個minimize過程的最後一個結構，且.coor檔的格式與pdb檔的格式一致，因此將drug\_H\_mini\_out.coor直接改名為drug\_H\_mini\_out.pdb即可，及為最佳化之結構，dcd檔為過程軌跡

**Protein PDB的處理**

**Docking**

使用MGLTools-1.5.6以及AutoDock Vina 1.1.2

=>先準備Vina所需要的protein的.pdbqt檔以及藥物的.pdbqt檔

Protein (accepter)

1. 開啟AutoDockTools-1.5.6
2. 開啟檔案，File=>Read Molecule=>選擇protein檔案
3. 去除結晶水，Edit=>Delete Water
4. 去除hydrogens，Edit=>Delete=>Delete Hydrogens
5. 加入Polar(O,N)上的氫，Edit=>Hydrogens=>Add=>Polar only(因為X-ray解不出氫，所以要把H補回去)
6. 設定Docking範圍，Grid=>Grid Box，設定Grid Box的大小及中心點，Spacing設為1，紀錄設定值
7. 儲存檔案，Grid=>Macromolecule=>Choose，點選要儲存的檔案=>Select Molecule=>Save protein.pdbqt，將檔案儲存為.pdbqt檔，為Vina需要的副檔名

Ligand (drug)

1. AutoDockTools-1.5.6
2. 開啟檔案，Ligand=>Input=>open，選擇要開啟的檔案
3. 選擇Docking可轉動的鍵結，Ligand=>Torsion Tree=> Choose Torsion (Green: rotatable, Magenta: non-rotatable, Red: unrotatable)=> Done
4. 儲存檔案，Ligand=>Output=>Save as PDBQT，將ligand儲存為.pdbqt檔

Autodock Vina

1. 設定vina所需要的檔案

開啟一空白文件=>

receptor = protein.pdbqt[[9]](#footnote-9)

ligand =drug\_H\_mini\_out.pdbqt[[10]](#footnote-10)

center\_x =

center\_y =

center\_z = [[11]](#footnote-11)

size\_x =

size\_y =

size\_z = [[12]](#footnote-12)

out = vina.out.pdbqt[[13]](#footnote-13)

log = vina.log.txt[[14]](#footnote-14)

exhaustiveness = 8[[15]](#footnote-15)

num\_modes = 9[[16]](#footnote-16)

energy\_range = 3[[17]](#footnote-17)

將檔案存為vina.conf.txt (可自行設定)

1. 在該資料夾開啟命令視窗，輸入指令=>vina –-config vina.conf.txt
2. 會產生vina.out.pdbqt以及vina.log.txt兩個輸出檔

vina.out.pdbqt: 包含所有結果的座標，只含Ligand不含Protein

vina.log.txt: 紀錄每個結果的affinity

1. 開啟命令視窗，輸入指令perl vinaPP.01.pl vina.out.pdbqt

vinaPP.01.pl: 可分析輸出檔vina.out.pdbqt的結果，將其存為多個pdb檔

vinaPP.01.pl

#!/bin/usr/perl

#vina post processing

#v.0.0.1-20100505

use strict;

if ($ARGV[0] eq "")

{

print "Please assign 1 parameters, when executing this script.\n";

print "\t1 the file name of input data.\m\n";

exit();

}

my $line;

my @words;

my @tmp;

open(INFILE, "$ARGV[0]");

while($line = <INFILE>)

{

chomp($line);

@words = split(/\s+/, $line);

if($words[0] eq "MODEL")

{

$tmp[0] = sprintf("%02d", $words[1]);

open(OUTFILE, "> $tmp[0].pdb");

while($line = <INFILE>)

{

@words = split(/\s+/, $line);

if($words[0] eq "ATOM")

{

print OUTFILE $line;

}

elsif($words[0] eq "ENDMDL")

{

print OUTFILE "END\n";

close OUTFILE;

last;

}

}

}

}

close INFILE;

**Complex**

將Protein (receptor)、Protein上的金屬離子以及Ligand組合為一Complex，並建立.psf和.pdb檔

開啟命令視窗，輸入vmd –e 02\_addHAndGetPSF\_AZX.txt

02\_addHAndGetPSF\_AZX.txt

package require psfgen

topology top\_all27\_prot\_na.txt

topology top\_all36\_cgenff\_400.txt

topology zinc\_finger\_top.txt

topology drug\_top.txt

pdbalias residue HIS HSE

pdbalias atom ILE CD1 CD

segment A {pdb protein.pdb;first nter;last cter}

segment Z {pdb zinc.pdb;first none;last none}

segment X {pdb 01.pdb;first none;last none}

patch ZNSP Z:401 A:193 A:196 A:228 A:230

coordpdb protein.pdb A

coordpdb zinc.pdb Z

coordpdb 01.pdb X

guesscoord

writepdb protein\_ligand\_zn\_01.pdb

writepsf protein\_ligand\_zn\_01.psf

exit

**建立水盒子以及添加離子**

開啟VMD讀取protein\_ligand\_zn\_01.pdb

建立水盒子，Extensions=>Modeling=>Add Solvate Box，勾選Rotate to minimize volume，設定水盒子邊界為8-10 Å=>Solvate

添加離子，Extensions=>Modeling=>Add Ions，勾選Neutralize and set NaCl concentration to 0.15 mol/L=>Autoionize

**進行分子動力學模擬**

* Minimize

結構能量最佳化，cellBasisVector為box大小，PmeGridsize大於box最好為2,3,5的倍數，讀取.psf與.pdb及parameter，constraints為限制，根據case的需求而定，進入機房，輸入指令namdsub 01\_mini.namd

01\_mini.namd

structure ionized.psf

coordinates ionized.pdb

temperature 0

paraTypeCharmm on

parameters par\_all27\_prot\_na.txt

parameters par\_all36\_cgenff\_400.txt

parameters zinc\_finger\_par.txt

parameters drug\_par.txt

outputEnergies 1000

outputTiming 1000

xstFreq 1000

dcdFreq 1000

wrapAll on

wrapNearest on

timestep 1

nonBondedFreq 2

fullElectFrequency 4

stepsPerCycle 20

switching on

switchDist 10

cutoff 12

pairlistdist 14

cellOrigin 0 0 0

cellBasisVector1 80.00 00.00 00.00

cellBasisVector2 00.00 80.00 00.00

cellBasisVector3 00.00 00.00 110.00

Pme on

PmeGridsizeX 80

PmeGridsizeY 80

PmeGridsizeZ 110

exclude scaled1-4

1-4scaling 1.0

binaryoutput off

outputname mini\_01\_out

restartname mini\_01\_rst

##########

constraints on

consexp 10

consref cons\_ref.pdb

conskfile cons\_ref.pdb

conskcol B

constraintScaling 0.1

##########

restartfreq 1000

minimize 50000

輸出檔為mini\_01\_out和mini\_01\_rst(副檔名含coor,dcd,vel,xsc,xst)檔案

* Heating

接續前步驟進行，讀取上步驟的檔案進行加熱，初始溫度為100K，每跑1000步加熱20K，直至溫度到300K，Bincoordinates, binvelocities, extendedSystem改為前一步的輸出檔檔名，outputname與restartname也要修改，總共跑1000000步(1ns)

02\_heat.namd

structure ../ionized.psf

coordinates ../ionized.pdb

bincoordinates ../mini\_01\_rst.coor

binvelocities ../mini\_01\_rst.vel

extendedSystem ../mini\_01\_rst.xsc

paraTypeCharmm on

parameters ../par\_all27\_prot\_na.txt

parameters ../par\_all36\_cgenff\_400.txt

parameters ../zinc\_finger\_par.txt

parameters ../drug\_par.txt

outputEnergies 1000

outputTiming 1000

xstFreq 1000

dcdFreq 1000

wrapAll on

wrapNearest on

timestep 1

nonBondedFreq 2

fullElectFrequency 4

stepsPerCycle 20

switching on

switchDist 10

cutoff 12

pairlistdist 14

Pme on

PmeGridsizeX 80

PmeGridsizeY 80

PmeGridsizeZ 110

exclude scaled1-4

1-4scaling 1.0

langevin on

langevinDamping 1

langevinTemp 300

langevinHydrogen on

langevinPiston on

langevinPistonTarget 1.01325

langevinPistonPeriod 200

langevinPistonDecay 100

langevinPistonTemp 300

binaryoutput off

outputname heat\_01\_our

restartname heat\_01\_rst

##########

constraints on

consexp 10

consref ../cons\_ref.pdb

conskfile ../cons\_ref.pdb

conskcol B

constraintScaling 0.1

##########

restartfreq 1000

run 0

langevinTemp 100

langevinPistonTemp 100

run 1000

langevinTemp 120

langevinPistonTemp 120

run 1000

langevinTemp 140

langevinPistonTemp 140

run 1000

langevinTemp 160

langevinPistonTemp 160

run 1000

langevinTemp 180

langevinPistonTemp 180

run 1000

langevinTemp 200

langevinPistonTemp 200

run 1000

langevinTemp 220

langevinPistonTemp 220

run 1000

langevinTemp 240

langevinPistonTemp 240

run 1000

langevinTemp 260

langevinPistonTemp 260

run 1000

langevinTemp 280

langevinPistonTemp 280

run 1000

langevinTemp 300

langevinPistonTemp 300

run 990000

* Equilibrium

將溫度固定在300K進行平衡，跑5000000步(5ns)

03\_equi.namd

structure ../../ionized.psf

coordinates ../../ionized.pdb

bincoordinates ../heat\_01\_rst.coor

binvelocities ../heat\_01\_rst.vel

extendedSystem ../equi\_01\_rst.xsc

paraTypeCharmm on

parameters ../../par\_all27\_prot\_na.txt

parameters ../../par\_all36\_cgenff\_400.txt

parameters ../../zinc\_finger\_par.txt

parameters ../../drug\_par.txt

outputEnergies 1000

outputTiming 1000

xstFreq 1000

dcdFreq 1000

wrapAll on

wrapNearest on

timestep 1

nonBondedFreq 2

fullElectFrequency 4

stepsPerCycle 20

switching on

switchDist 10

cutoff 12

pairlistdist 14

Pme on

PmeGridsizeX 80

PmeGridsizeY 80

PmeGridsizeZ 110

exclude scaled1-4

1-4scaling 1.0

langevin on

langevinDamping 1

langevinTemp 300

langevinHydrogen on

langevinPiston on

langevinPistonTarget 1.01325

langevinPistonPeriod 200

langevinPistonDecay 100

langevinPistonTemp 300

binaryoutput off

outputname equi\_01\_out

restartname equi\_01\_rst

##########

constraints on

consexp 10

consref ../../cons\_ref.pdb

conskfile ../../cons\_ref.pdb

conskcol B

constraintScaling 0.1

##########

restartfreq 1000

run 5000000

總共跑5000000步(5ns)，再接續跑5000000步(5ns)，總共跑10000000步(10ns)

04\_equi.namd

structure ../../../ionized.psf

coordinates ../../../ionized.pdb

bincoordinates ../equi\_01\_rst.coor

binvelocities ../equi\_01\_rst.vel

extendedSystem ../equi\_01\_rst.xsc

paraTypeCharmm on

parameters ../../../par\_all27\_prot\_na.txt

parameters ../../../par\_all36\_cgenff\_400.txt

parameters ../../../zinc\_finger\_par.txt

parameters ../../../drug\_par.txt

outputEnergies 1000

outputTiming 1000

xstFreq 1000

dcdFreq 1000

wrapAll on

wrapNearest on

timestep 1

nonBondedFreq 2

fullElectFrequency 4

stepsPerCycle 20

switching on

switchDist 10

cutoff 12

pairlistdist 14

Pme on

PmeGridsizeX 80

PmeGridsizeY 80

PmeGridsizeZ 110

exclude scaled1-4

1-4scaling 1.0

langevin on

langevinDamping 1

langevinTemp 300

langevinHydrogen on

langevinPiston on

langevinPistonTarget 1.01325

langevinPistonPeriod 200

langevinPistonDecay 100

langevinPistonTemp 300

binaryoutput off

outputname equi\_equi\_01\_out

restartname equi\_equi\_01\_rst

##########

constraints on

consexp 10

consref ../../../cons\_ref.pdb

conskfile ../../../cons\_ref.pdb

conskcol B

constraintScaling 0.1

##########

restartfreq 1000

firstTimeStep 5000000

run 5000000

dcd檔為軌跡檔，將生成的.dcd檔存回電腦做後續的分析，若沒有平衡、RMSD沒有穩定，則可以繼續跑平衡

**分析結果**

將dcd檔存至電腦後，開啟VMD，先讀取與其對應的psf檔=> ionized.psf(含水)，再讀取dcd檔，若無法一次讀取dcd檔，可將dcd檔每1000個frame去水儲存為一個dcd檔，分段讀取分析

分段步驟：

1. File=>New Molecule=>Browse: 先讀取相對應的PSF檔=>Load
2. Browse: 讀取欲分析的dcd檔=>first:0 Last:999 Stride:1，從第0個讀到第999個frame，總共會有1000個frame=>Load
3. 再將讀進去的frame去水儲存，File=>Save Coordinates

=>Selected atoms: chain A Z X=>File type: dcd=>Save，存檔類型選擇dcd

1. 點選該項目(Select molecule)=>右鍵=>Delete Frames=>Delete
2. 重複step 2 ~ step 4，更改first & Last，每1000個frame存一次即可

關掉VMD再重開，先讀取去水的psf，再將已分段去水的dcd照順序讀進來，另存為一個新的、完整的dcd檔

RMSD:

用來判斷結構是否已經達到平衡，Extensions=>Analysis=>RMSD Trajectory Tool=>輸入protein=>點選backbone進行ALIGN，會以protein為中心固定backbone，再將protein改為chain X，Selection Modifiers選擇noh，點選RMSD，可觀察chain X(藥物)是否趨於穩定，勾選Plot會產生圖，勾選Save可儲存RMSD分析數據，若有達到平衡，則RMSD的值會趨於穩定

#####使用程式只看藥物的RMSD

**自由能計算**

Δ*G* = Δ*Gcomplex*– (Δ*G*enzyme + Δ*G*substrate )

分為MM、NMA、APBS三步=> Δ*GX =* Δ*EMM* (MM)+ Δ*G*solv (APBS) – *T*Δ*S* (NMA)

根據RMSD判斷，挑選軌跡檔中較穩定的一段來計算，長度大約5ns至10ns，將該區dcd檔另外存出，frame數調整為1001個frame

需將軌跡檔分為complex、protein(包含蛋白質上的金屬)、ligand三部分，也需要準備complex、protein、ligand的pdb與psf (不含水, 跑前跑後都可以)

將dcd檔分開:

讀取已去水且RMSD較穩定的一段dcd，

File=>Save Coordinates=>Selected atoms: chain A Z (只有protein和離子) =>save

File=>Save Coordinates=>Selected atoms: chain X (只有藥物) =>save

* **MM : Molecular mechanics**

將complex、protein、ligand分別計算，需要讀取psf、pdb、rst.xsc檔(平衡任一部分都可)以及dcd，將檔案放至機房使用namdsub mm送job即可

**Complex:**

structure protein\_ligand\_zn\_01.psf

coordinates protein\_ligand\_zn\_01.pdb

extendedSystem equi\_equi\_01\_rst.xsc

paraTypeCharmm on

parameters par\_all27\_prot\_na.txt

parameters par\_all36\_cgenff\_400.txt

parameters drug\_par.txt

parameters zinc\_finger\_par.txt

temperature 0

timestep 1

nonBondedFreq 1

fullElectFrequency 1

stepsPerCycle 1

switching on

switchDist 10

cutoff 12

pairlistdist 14

Pme on

PmeGridsizeX 80

PmeGridsizeY 80

PmeGridsizeZ 110

exclude scaled1-4

binaryoutput off

outputname inter\_out

coorfile open dcd equi\_equi\_01\_out\_chainazx.dcd

coorfile read

set ts 0

firstTimestep $ts

run 0

while {[coorfile read] == 0} {

incr ts 10

#unit: 1ps

firstTimestep $ts

run 0

}

coorfile close

**Protein:**

structure protein\_zn\_01.psf

coordinates protein\_zn\_01.pdb

extendedSystem equi\_equi\_01\_rst.xsc

paraTypeCharmm on

parameters par\_all27\_prot\_na.txt

parameters par\_all36\_cgenff\_400.txt

parameters drug\_par.txt

parameters zinc\_finger\_par.txt

temperature 0

timestep 1

nonBondedFreq 1

fullElectFrequency 1

stepsPerCycle 1

switching on

switchDist 10

cutoff 12

pairlistdist 14

Pme on

PmeGridsizeX 80

PmeGridsizeY 80

PmeGridsizeZ 110

exclude scaled1-4

binaryoutput off

outputname inter\_out

coorfile open dcd equi\_equi\_01\_out\_chainaz.dcd

coorfile read

set ts 0

firstTimestep $ts

run 0

while {[coorfile read] == 0} {

incr ts 10

#unit: 1ps

firstTimestep $ts

run 0

}

coorfile close

**Ligand:**

structure ligand\_01.psf

coordinates ligand\_01.pdb

extendedSystem equi\_equi\_01\_rst.xsc

paraTypeCharmm on

parameters par\_all27\_prot\_na.txt

parameters par\_all36\_cgenff\_400.txt

parameters drug\_par.txt

parameters zinc\_finger\_par.txt

temperature 0

timestep 1

nonBondedFreq 1

fullElectFrequency 1

stepsPerCycle 1

switching on

switchDist 10

cutoff 12

pairlistdist 14

Pme on

PmeGridsizeX 80

PmeGridsizeY 80

PmeGridsizeZ 110

exclude scaled1-4

binaryoutput off

outputname inter\_out

coorfile open dcd equi\_equi\_01\_out\_chainx.dcd

coorfile read

set ts 0

firstTimestep $ts

run 0

while {[coorfile read] == 0} {

incr ts 10

#unit: 1ps

firstTimestep $ts

run 0

}

coorfile close

會得到mm\_out、mm\_out.coor、mm\_out.vel、mm\_out.xsc檔，開啟mm\_out檔，需要ELECT以及VDW的值



執行perl namdEnergy.pl mm.out 將值抓出來，輸出的檔為mm\_out.1與mm\_out.10，分別為每一個frame的數值以及每十個為單位的數值

namdEnergy.pl

open(IN, "$ARGV[0]");

open(OUT, ">$ARGV[0].1");

open(OUTT, ">$ARGV[0].10");

$i = 0;

while($line = <IN>)

{

if($line =~ /ENERGY: /)

{

print OUT $line;

if($i % 10 == 0)

{

print OUTT $line;

}

$i++;

}

}

close IN;

close OUT;

※Elect的值會重複算兩次，因此在excel輸入數值時需除二

**NMA : Normal-mode analysis**

將complex、protein、ligand分別計算，因每個frame都要計算會算得比較久，固可將dcd檔的1001個frame轉存為101個frame計算

執行vmd –e 01makePDBFromDCD.txt，將每一個frame生成一個pdb檔，再執行vmd –e 02seprateMultipleChains.txt，將每個chain產生不同的pdb檔

01makePDBFromDCD.txt

mol load psf protein\_ligand\_zn\_01.psf dcd equi\_equi\_01\_out\_chainazx.dcd

set all [atomselect top all]

for {set i 0} {$i < 101} {set i [expr $i + 1]} {

animate write pdb equi\_equi\_01\_out\_chainazx.$i.pdb beg $i end $i sel $all top

}

exit

02seprateMultipleChains.txt

mol load psf protein\_ligand\_zn\_01.psf dcd equi\_equi\_01\_out\_chainazx.dcd

set chainA [atomselect top "chain A"]

$chainA writepdb protein\_ligand\_zn\_01\_chaina.pdb

set chainZ [atomselect top "chain Z"]

$chainZ writepdb protein\_ligand\_zn\_01\_chainz.pdb

set chainX [atomselect top "chain X"]

$chainX writepdb protein\_ligand\_zn\_01\_chainx.pdb

exit

將產生出的檔案送至機房，執行perl 04mkALotDir.pl，04mkALotDir.pl會直接使用03charmmCoorForPSF.pl讀取pdb檔，生成charmm計算所需要的檔案，03charmmCoorForPSF.pl不需要更改，04mkALotDir.pl需要根據complex、protein、ligand去修改

03charmmCoorForPSF.pl

#!/usr/bin/perl

# v0.0.1-20070517

#Written by Sanskrit Gao

use strict;

my $line;

my @line1;

my @line2;

my $n;

open INPDB, "@ARGV[0]";

open OUTPDB, "> @ARGV[1]";

$n = 0;

while($line = <INPDB>)

{

chomp($line);

@line2 = split(/\s+/, $line);

if(@line2[0] ne "ATOM")

{

print OUTPDB "$line\n";

}

else

{

@line2[0] = substr($line, 0, 22);

@line2[5] = " ".substr($line, 22, 4);

if(@line2[5] != @line1[5])

{

$n++;

}

if($n < 10)

{

@line2[0] .= " $n";

}

elsif($n < 100)

{

@line2[0] .= " $n";

}

elsif($n < 1000)

{

@line2[0] .= " $n";

}

elsif($n < 10000)

{

@line2[0] .= "$n";

}

else

{

print "Too many residues (number > 9999).\n";

exit;

}

@line2[0] .= substr($line, 26);

@line2[0] .= "\n";

print OUTPDB "@line2[0]";

}

@line1 = split(/\s+/, $line);

@line1[5] = @line2[5];

}

close INPDB;

close OUTPDB;

04mkALotDir.pl

#!/usr/bin/perl

my $charmmName = "/home/bin/charmm35b1/charmmHugeI";

my $i;

system "mkdir results";

$i = 0;

while($i < 101)

{

system "perl 03charmmCoorForPSF.pl equi\_equi\_01\_out\_chainazx.$i.pdb equi\_equi\_01\_out\_chainazx.$i.pdb.ok";

open(OUT, "> 03buildCharmmPSF.$i.txt");

print OUT 'BOMLEV -1'."\n";

print OUT 'OPEN READ UNIT 21 CARD NAME "top\_all27\_prot\_na.txt"'."\n";

print OUT 'READ RTF UNIT 21 CARD'."\n";

print OUT 'CLOSE UNIT 21'."\n";

print OUT 'OPEN READ UNIT 21 CARD NAME "par\_all27\_prot\_na.txt"'."\n";

print OUT 'READ PARA flex UNIT 21 CARD'."\n";

print OUT 'CLOSE UNIT 21'."\n\n";

print OUT 'OPEN READ UNIT 21 CARD NAME "top\_all36\_cgenff.txt"'."\n";

print OUT 'READ RTF UNIT 21 CARD APPE'."\n";

print OUT 'CLOSE UNIT 21'."\n";

print OUT 'OPEN READ UNIT 21 CARD NAME "par\_all36\_cgenff.txt"'."\n";

print OUT 'READ PARA flex UNIT 21 CARD APPE'."\n";

print OUT 'CLOSE UNIT 21'."\n\n";

print OUT 'stream "drug.str"'."\n\n";

print OUT 'stream "zinc\_finger.str"'."\n\n";

print OUT 'open unit 21 card read name "protein\_ligand\_zn\_01\_chaina.pdb"'."\n";

print OUT 'read sequ pdb unit 21'."\n";

print OUT 'generate A setu'."\n";

print OUT 'close unit 21'."\n\n";

print OUT 'open unit 21 card read name "protein\_ligand\_zn\_01\_chainz.pdb"'."\n";

print OUT 'read sequ pdb unit 21'."\n";

print OUT 'generate Z setu'."\n";

print OUT 'close unit 21'."\n\n";

print OUT 'open unit 21 card read name "protein\_ligand\_zn\_01\_chainx.pdb"'."\n";

print OUT 'read sequ pdb unit 21'."\n";

print OUT 'generate X setu'."\n";

print OUT 'close unit 21'."\n\n";

print OUT 'patch ZNSP Z 401 A 193 A 196 A 228 A 230'."\n\n";

print OUT 'open unit 21 card read name "equi\_equi\_01\_out\_chainazx.'."$i".'.pdb.ok"'."\n";

print OUT 'read coor pdb unit 21'."\n";

print OUT 'close unit 21'."\n\n";

print OUT 'hbuild sele hydrogen end'."\n\n";

print OUT 'ic fill'."\n";

print OUT 'ic para'."\n";

print OUT 'ic build'."\n\n";

print OUT 'print coor'."\n\n";

print OUT 'open write formatted unit 21 name "equi\_equi\_01\_out\_chainazx.charmm.psf"'."\n";

print OUT 'write psf card unit 21'."\n";

print OUT '\* PSF'."\n";

print OUT '\*'."\n";

print OUT 'close unit 21'."\n\n";

print OUT 'open unit 21 card write name "equi\_equi\_01\_out\_chainazx.'."$i".'.charmm.crd"'."\n";

print OUT 'write coor card unit 21'."\n";

print OUT 'close unit 21'."\n\n";

print OUT 'stop'."\n";

close OUT;

open(OUT, "> nma.$i.charmm");

print OUT 'BOMLEV -1'."\n";

print OUT 'OPEN READ UNIT 21 CARD NAME "top\_all27\_prot\_na.txt"'."\n";

print OUT 'READ RTF UNIT 21 CARD'."\n";

print OUT 'CLOSE UNIT 21'."\n";

print OUT 'OPEN READ UNIT 21 CARD NAME "par\_all27\_prot\_na.txt"'."\n";

print OUT 'READ PARA flex UNIT 21 CARD'."\n";

print OUT 'CLOSE UNIT 21'."\n\n";

print OUT 'OPEN READ UNIT 21 CARD NAME "top\_all36\_cgenff.txt"'."\n";

print OUT 'READ RTF UNIT 21 CARD APPE'."\n";

print OUT 'CLOSE UNIT 21'."\n";

print OUT 'OPEN READ UNIT 21 CARD NAME "par\_all36\_cgenff.txt"'."\n";

print OUT 'READ PARA flex UNIT 21 CARD APPE'."\n";

print OUT 'CLOSE UNIT 21'."\n\n";

print OUT 'stream "drug.str"'."\n\n";

print OUT 'stream "zinc\_finger.str"'."\n\n";

print OUT 'open read formatted unit 11 name "equi\_equi\_01\_out\_chainazx.charmm.psf"'."\n";

print OUT 'read psf card unit 11'."\n";

print OUT 'close unit 11'."\n";

print OUT 'open read formatted unit 11 name "equi\_equi\_01\_out\_chainazx.'."$i".'.charmm.crd"'."\n";

print OUT 'read coor card unit 11'."\n";

print OUT 'close unit 11'."\n";

print OUT 'NBON INBF -1 ELEC NOEW NOFMA ATOM CDIE FSWI VDW VATOM VSWI CUTNB 10.0 CTOFNB 8.0 CTONNB 6.0'."\n";

print OUT 'energy'."\n";

print OUT '!'."\n";

print OUT '! SD minimization 500 steps'."\n";

print OUT '! --------------------------'."\n";

print OUT 'open write unit 22 file name "equi\_equi\_01\_out\_chainazx.'."$i".'.minied.sd.charmm.dcd"'."\n";

print OUT 'date'."\n";

print OUT 'mini sd nsteps 500 INBF -1 ELEC NOEW NOFMA ATOM CDIE FSWI VDW VATOM VSWI nprint 100 TOLGRD 0.00000001 iuncrd 22 nsavc 100'."\n";

print OUT 'date'."\n";

print OUT '!'."\n";

print OUT '! CG minimization to TOLGRD to 1E-3'."\n";

print OUT '! ---------------------------------'."\n";

print OUT 'open write unit 22 file name "equi\_equi\_01\_out\_chainazx.'."$i".'.minied.cg.charmm.dcd"'."\n";

print OUT 'date'."\n";

print OUT 'mini conj nsteps 100000000 INBRFQ -1 IHBRFQ -1 NBON INBF -1 ELEC NOEW NOFMA ATOM CDIE FSWI VDW VATOM VSWI nprint 1000 TOLGRD 0.001 iuncrd 22 nsavc 1000'."\n";

print OUT 'date'."\n";

print OUT '!'."\n";

print OUT '! ABNR minimization to TOLGRD to 1E-6'."\n";

print OUT '! -----------------------------------'."\n";

print OUT 'open write unit 22 file name "equi\_equi\_01\_out\_chainazx.'."$i".'.minied.abnr.charmm.dcd"'."\n";

print OUT 'date'."\n";

print OUT 'mini abnr nsteps 100000000 INBF -1 ELEC NOEW NOFMA ATOM CDIE FSWI VDW VATOM VSWI nprint 1000 TOLGRD 0.000001 iuncrd 22 nsavc 1000'."\n";

print OUT 'date'."\n";

print OUT '!'."\n";

print OUT 'BOMLEV -2'."\n";

print OUT '!'."\n";

print OUT '! noh Regular diagonalization 5056\*3 = 15168'."\n";[[18]](#footnote-18)

print OUT '! -----------------------------------------'."\n";

print OUT 'VIBRan NMODes 15168'."\n";

print OUT 'DIAGonalize ENTRopy TEMP 300.0'."\n";

print OUT 'print norm'."\n";

print OUT 'ther'."\n";

# print OUT 'SET 1 1'."\n";

# print OUT 'label loop2'."\n";

# print OUT 'FLUC ATOM sele resid @1 .and. segid A end'."\n";

# print OUT 'incr 1 by 1'."\n";

# print OUT 'if 1 LT 262 goto loop2'."\n";

print OUT 'end'."\n";

print OUT 'date'."\n";

print OUT 'stop'."\n";

close OUT;

system "mkdir $i;mv \*.$i.\* $i/";

system "cp protein\_ligand\_zn\_01\_\*.pdb top\_\*.txt par\_\*.txt \*.str $i/ ";

system "cd $i;$charmmName < 03buildCharmmPSF.$i.txt > 03buildCharmmPSF.$i.log.txt";

system "mv $i results/";

$i++;

}

此步驟執行後會產生results資料夾，內含有與frame數相等的資料夾，每個資料夾都含有topology、parameter、每一個frame的pdb、每一個chain的pdb、crd、pdb.ok、psf、03buildCharmmPSF.$i.txt、03buildCharmmPSF.$i.log.txt、nma.$i.charmm，需開啟每一個03buildCharmmPSF.$i.log.txt觀察是否有錯誤，若有正常結束會出現NORMAL TERMINATION，可進入results資料夾輸入grep 'NORMAL TERMINATION' \*/\*.log.txt，就可觀察到是否每個軌跡都有正常結束

將submitJob.pl放在results資料夾下，執行後可將多個檔案送出計算

submitJob.pl

#!/usr/bin/perl

my $i;

open(OUT, "> qset");

print OUT "3-4g\nNS5\n";

close OUT;

$i = 0;

while($i < 101)

{

system "cp qset $i;cd $i;charmmsub nma.$i < qset;rm qset";

$i++;

}

跑完以後將getlog.pl、readCharmmNMALog.pl、readCharmmNMALogForEnergy.pl三個檔案放至results資料夾中，執行perl getlog.pl，會將資料夾整理成放置nma.$i.out的bigOutput與放置所有input的smallLog資料夾，並且產生記錄所有能量的energy.log.txt檔案

getlog.pl

#!/usr/bin/perl

use strict;

my $i;

$i = 0;

system "cat > energy.log.txt <<EOF\nTS(rot) -TS(tra) -TS(vib) -TS(total) G H -TS(vib) kcal/mol\nEOF";

while($i < 101)

{

system "cd $i;perl ../readCharmmNMALog.pl nma.$i.out nma.$i.01.needed.txt nma.$i.01.frequenies.dat;perl ../readCharmmNMALogForEnergy.pl nma.$i.01.needed.txt nma.$i.01.energy.txt";

system "cat $i/nma.$i.01.energy.txt >> energy.log.txt";

system "cat >> energy.log.txt <<EOF\n\nEOF";

$i += 1;

}

$i = 0;

system "mkdir bigOutput";

system "mkdir smallLog";

while($i < 101)

{

system "mv $i/nma.$i.out bigOutput/";

system "mv $i smallLog/";

$i += 1;

}

readCharmmNMALog.pl

#!/usr/bin/perl

#Sanskrit Gao

#v.20070705

use strict;

my $entropyLine = 10;

my $modeNumber = "Nine";

my $modeLine = 45;# $modeNumber \* 5

my $thermoLine = 26;

my $line;

my @line2;

my $i;

if (@ARGV[0] eq "" || @ARGV[1] eq ""|| @ARGV[2] eq "")

{

print "Please assign one input and two ouput file names, when executing this script.";

exit();

}

open IN, "@ARGV[0]";

open OUT, "> @ARGV[1]";

open OUT2, "> @ARGV[2]";

while($line = <IN>)

{

chomp($line);

@line2 = split(/\s+/, $line);

if($line eq " ENTROPY>")

{

$i = 0;

print OUT "====== Entorpy ======\n";

while($i < $entropyLine)

{

print OUT "$line\n";

$line = <IN>;

chomp($line);

$i++;

}

print OUT "\n";

}

elsif(@line2[1] eq "FREQUENCIES")

{

$line = <IN>;

$line = <IN>;

chomp($line);

@line2 = split(/\s+/, $line);

while(@line2[1] ne "Principal" && @line2[1] ne "\*\*\*\*" && @line2[1] ne "")

{

@line2 = split(/\s+/, $line);

$i = 1;

while(@line2[$i] ne "")

{

print OUT2 "@line2[$i] ";

$i++;

print OUT2 "@line2[$i]\n";

$i++;

}

$line = <IN>;

chomp($line);

@line2 = split(/\s+/, $line);

}

}

elsif(@line2[1] eq "VIBRATION")

{

$i = 0;

print OUT "====== The First $modeNumber Mode ======\n";

while($i < $modeLine)

{

print OUT "$line\n";

$line = <IN>;

chomp($line);

$i++;

}

while(($line = <IN>) && $line ne " VIBRAN> SET 1 1\n")

{

chomp($line);

if($line eq " VIBRAN> ther")

{

$i = 0;

print OUT "====== Thermodynamics ======\n";

while($i < $thermoLine)

{

print OUT "$line\n";

$line = <IN>;

chomp($line);

$i++;

}

print OUT "\n";

print OUT "====== Fluctuation of Every Residue ======\n";

}

}

}

elsif((@line2[1] eq "VIBRAN>") && @line2[2] eq "label")

{

$line = <IN>;

$line = <IN>;

@line2 = split(/\s+/, $line);

print OUT "yoyoyoyoyo\n";

print OUT "@line2[8] @line2[9]\n";

open OUT3, "> @line2[9].fluctuation.dat";

$i = 0;

while($line ne " IF test evaluated as false. Skipping command")

{

if((@line2[1] eq "AVERAGE") && @line2[2] eq "FLUCTUATION")

{

$i++;

print OUT3 "$i @line2[7]\n";

print OUT "@line2[7]\n";

}

$line = <IN>;

@line2 = split(/\s+/, $line);

chomp($line);

}

}

}

close IN;

close OUT;

close OUT2;

close OUT3;

readCharmmNMALogForEnergy.pl

#!/usr/bin/perl

#Sanskrit Gao

#v.0.0.2-20080307

#v.20070704

#readCharmmNMALogForEnergy.pl nma.01.needed.txt nma.01.energy.txt

use strict;

my $line;

my @line2;

my $i;

my $negTS;

my @ii;

open IN, "@ARGV[0]";

open OUT, "> @ARGV[1]";

$i = 0;

while($i < 5)

{

$line = <IN>;

$i++;

}

@line2 = split(/\s+/, $line);

$ii[0] = $line2[4] / 1000 \* (-1);#$ii[0]:-T(cal=>kcal)

while($i < 9)

{

$line = <IN>;

$i++;

}

@line2 = split(/\s+/, $line);

$ii[1] = $line2[3] \* $ii[0];#$ii[1]:-TS(rot)

$line = <IN>;

$i++;

@line2 = split(/\s+/, $line);

$ii[2] = $line2[3] \* $ii[0];#$ii[2]:-TS(tra)

$line = <IN>;

$i++;

@line2 = split(/\s+/, $line);

$ii[3] = $line2[3] \* $ii[0];#$ii[3]:-TS(vib)

$line = <IN>;

$i++;

$ii[4] = $ii[1] + $ii[2] + $ii[3];

while($i < 83)

{

$line = <IN>;

$i++;

}

chomp($line);

@line2 = split(/\s+/, $line);

$negTS = @line2[4]\*300\*(-1);

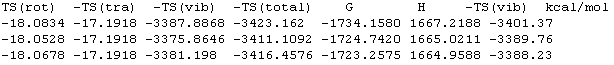
# print OUT "-TS(rot) -TS(tra) -TS(vib) -TS(total) G H -TS(vib) kcal/mol\n";

print OUT "$ii[1] $ii[2] $ii[3] $ii[4] @line2[2] @line2[5] $negTS";

close IN;

close OUT;

開啟energy.log.txt如下圖，結果為”-TS(total)”



再來把值由101個擴充為1001個，將抓下來的-TS值另存新檔為NMA\_pre，執行perl forG.pl NMA\_pre.txt NMA.txt，在手動刪除第一組相同數值五個，以及最後一組相同數值四個

forG.pl

open(IN, "$ARGV[0]");

open(OUT, "> $ARGV[1]");

while($line = <IN>)

{

for($i = 0; $i < 10; $i++)

{

print OUT "$line";

}

}

close IN;

close OUT;

**APBS : Adaptive Poisson-Boltzmann Solver**

分成polar及apolar，再分為complex、protein與ligand三部分

polar需要準備的檔案為=>

polar.in、1001個frame的pdb、run.01.sh、CHARMM.DAT、exeApbs01.pl、getApbsEnergy01.pl

apolar需要準備的檔案為=>

apolar.in、1001個frame pdb、run.01.sh、vparam-amber-parm94.sans.20100422.dat、exeApbs01.pl、getApbsEnergy01.pl

* **Polar**

執行vmd -e 01makePDBFromDCD.txt使每一個frame產生一個pdb檔，再執行 vmd –e 02findBoundary.txt > 02findBoundary.out，根據02findBoundary.out的內容更改polar.in檔

01makePDBFromDCD.txt

mol load psf protein\_ligand\_zn\_01\_new.psf dcd equi\_equi\_01\_out\_chainazx.dcd

set all [atomselect top all]

for {set i 0} {$i < 1001} {set i [expr $i + 1]} {

animate write pdb equi\_equi\_01\_out\_chainazx.$i.pdb beg $i end $i sel $all top

}

exit

02findBoundary.txt

mol load psf protein\_ligand\_zn\_01.psf dcd equi\_equi\_01\_out\_chainazx.dcd

set maxx 0.0

set maxy 0.0

set maxz 0.0

set minx 0.0

set miny 0.0

set minz 0.0

set frameNumber [molinfo top get numframes]

for {set i 0} { $i < $frameNumber } { incr i } {

set allAtom [atomselect top "all" frame $i]

foreach xx [$allAtom get x] {

if {$xx > $maxx} {

set maxx $xx

} elseif {$xx < $minx} {

set minx $xx

}

}

foreach yy [$allAtom get y] {

if {$yy > $maxy} {

set maxy $yy

} elseif {$yy < $miny} {

set miny $yy

}

}

foreach zz [$allAtom get z] {

if {$zz > $maxz} {

set maxz $zz

} elseif {$zz < $minz} {

set minz $zz

}

}

}

set xcen [expr ($maxx + $minx) / 2]

set ycen [expr ($maxy + $miny) / 2]

set zcen [expr ($maxz + $minz) / 2]

set xdis [expr $maxx - $minx]

set ydis [expr $maxy - $miny]

set zdis [expr $maxz - $minz]

puts "x min: $minx max: $maxx center: $xcen distance: $xdis"

puts "y min: $miny may: $maxy center: $ycen distance: $ydis"

puts "z min: $minz maz: $maxz center: $zcen distance: $zdis"

exit

根據02findBoundary.out的內容更改polar.in檔中的cglen、fglen、cgcent、fgcent，cgcent、fgcent=>為center，fglen=>為系統的最長邊+4，cglen=>為fglen x1.2，而dime要比cglen大

polar.in

#To caculate human Nit2-3ppa complex polar solvation free energy

read

end

elec

mg-auto

# 257 257 257 and atom number => ~3.8GiB

# 65 65 65 and atom number => ~210MiB

dime 129 129 129

# fglen \* 1.2 =~ cglen

cglen 112 112 112

fglen 93 93 93

cgcent 3.6498 9.0313 -5.2417

fgcent 3.6498 9.0313 -5.2417

mol 1

npbe

bcfl mdh

pdie 2.0000

sdie 78.5400

ion 1 0.150 1.76375

ion -1 0.150 2.27

srfm spl4

chgm spl4

sdens 10.00

srad 1.40

swin 0.30

temp 300

calcenergy total

calcforce no

end

elec

mg-auto

dime 129 129 129

cglen 112 112 112

fglen 93 93 93

cgcent 3.6498 9.0313 -5.2417

fgcent 3.6498 9.0313 -5.2417

mol 1

npbe

bcfl mdh

pdie 2.0000

sdie 2.0000

srfm spl4

chgm spl4

sdens 10.00

srad 1.40

swin 0.30

temp 300

calcenergy total

calcforce no

end

print elecEnergy 1 end

print elecEnergy 2 end

print elecEnergy 1 - 2 end

quit

將CHARMM.DAT自機房的home/bin/apbs1421/pdb2pqr/dat中載下來，加入藥物的資訊後再上傳至原來的地方，CHARMM.DAT格式如下，欄位依序為藥物名稱(residue name)、atom name、charge、atom大小相關數值(位在parameter檔的nonbonded的第四欄，van der Waals radius in A) 、atom type

#M16

M16 C1 0.427 1.8600 CG2RC0

M16 C2 0.309 1.8600 CG2RC0

M16 C3 0.088 2.0200 CG3C52

M16 C4 -0.208 2.0100 CG321

M16 C5 0.264 2.2000 CG2R53

將polar.in、1001個frame的pdb、run.01.sh、exeApbs01.pl、getApbsEnergy01.pl檔案上傳至機房

run.01.sh

source apbsOn

fileName="equi\_equi\_01\_out\_chainazx."

date

perl exeApbs01.pl $fileName 0 1 1000 .pdb "pdb2pqr.py" "--ffout=CHARMM --assign-only --ff=CHARMM" apbs polar.in

date

perl getApbsEnergy01.pl $fileName 0 1 1000 .log.txt polar 2

date

exeApbs01.pl

#!/usr/bin/perl

#exeApbs01.pl

#Sanskrit Gao

#v.0.0.2-20160325

use strict;

#@ARGV

#0 file name before counter. Ex: homo.Nit2.H.dyna\_out.01.

#1 first counter number. Ex: 101

#2 delta counter number. This can not less then 1. Ex: 1 (if 101 102 103...), 2 (if 101 103 105)

#3 last counter number.

#4 file name after counter. Ex: .pdb

#5 pdb2pqr full path. Ex: "/home/sanskrit/bin/pdb2pqr-1.2.1/pdb2pqr.py"

#6 pdb2pqr parameter. Ex: "--ffout=CHARMM.sans --apbs-input --assign-only --ff=CHARMM.sans"

#7 apbs full path or just apbs. Ex: apbs

#8 apbs input file name or full path. Ex: "/home/sanskrit/Desktop/ttt/homo.in"

if ($ARGV[0] eq "" || $ARGV[1] eq "" || $ARGV[2] eq "" || $ARGV[3] eq "" || $ARGV[4] eq "" || $ARGV[5] eq "" || $ARGV[6] eq "" || $ARGV[7] eq "" || $ARGV[8] eq "" )

{

print "Please assign 9 parameters, when executing this script.\n";

print "\t1 file name before the counter.\n";

print "\t2 first counter number.\n";

print "\t3 delta counter number. This can not less then 1.\n";

print "\t4 last counter number.\n";

print "\t5 file name after the counter.\n";

print "\t6 pdb2pqr full path\n";

print "\t7 pdb2pqr parameter.\n";

print "\t8 apbs full path or just apbs.\n";

print "\t9 apbs input file name or full path.\n\n";

exit();

}

my $line;

my @tmp;

while($ARGV[1] <= $ARGV[3])

{

open IN, "$ARGV[0]"."$ARGV[1]"."$ARGV[4]";

open OUT, ">tmp.data";

while($line = <IN>)

{

if($line =~ /^ATOM/ || $line =~ /^END/)

{

print OUT "$line";

}

}

close IN;

close OUT;

system "mv tmp.data $ARGV[0]"."$ARGV[1]"."$ARGV[4]";

system "$ARGV[5] $ARGV[6] $ARGV[0]"."$ARGV[1]"."$ARGV[4]"." $ARGV[0]"."$ARGV[1]".".pqr";

open IN, "$ARGV[0]"."$ARGV[1]".".pqr";

open OUT, ">tmp.data";

while($line = <IN>)

{

if($line =~ /^ATOM/)

{

$tmp[0] = substr($line, 0, 16);

$tmp[1] = substr($line, 16);

$line = "$tmp[0] $tmp[1]";

}

print OUT "$line";

}

close IN;

close OUT;

system "mv tmp.data $ARGV[0]"."$ARGV[1]".".pqr";

open IN, "$ARGV[8]";

open OUT, ">tmp.data";

while($line = <IN>)

{

if($line eq "read\n")

{

print OUT "$line";

print OUT " mol pqr $ARGV[0]"."$ARGV[1]".".pqr\n";

$line = <IN>;

}

print OUT "$line";

}

close IN;

close OUT;

system "mv tmp.data $ARGV[0]"."$ARGV[1]".".in";

system "$ARGV[7] --output-file=$ARGV[0]"."$ARGV[1]".".out.txt $ARGV[0]"."$ARGV[1]".".in > $ARGV[0]"."$ARGV[1]".".log.txt";

$ARGV[1] += $ARGV[2];

}

getApbsEnergy01.pl

#!/usr/bin/perl

#getApbsEnergy01.pl

#Sanskrit Gao

#v.0.0.2-20071213

#v.0.0.1-20070910

use strict;

#@ARGV

#0 file name before counter. Ex: homo.Nit2.H.dyna\_out.01.

#1 first counter number. Ex: 101

#2 delta counter number. This can not less then 1. Ex: 1 (if 101 102 103...), 2 (if 101 103 105)

#3 last counter number.

#4 file name after counter. Ex: .log.txt

#5 energy type. [polar, apolar, coulomb]

#6 epsilon of protein. Ex: 2

if ($ARGV[0] eq "" || $ARGV[1] eq "" || $ARGV[2] eq "" || $ARGV[3] eq "" || $ARGV[4] eq "" || $ARGV[5] eq "" || $ARGV[6] eq "")

{

print "Please assign 7 parameters, when executing this script.\n";

print "\t1 file name before the counter.\n";

print "\t2 first counter number.\n";

print "\t3 delta counter number. This can not less then 1.\n";

print "\t4 last counter number.\n";

print "\t5 file name after the counter.\n";

print "\t6 energy type. [polar, apolar, coulomb]\n";

print "\t7 epsilon of protein.\n\n";

exit();

}

my $line;

my @tmp;

my $tmp2;

my $sum;

my $num;

my $i;

$sum = 0;

$num = (($ARGV[3] - $ARGV[1]) / $ARGV[2]) + 1;

if($ARGV[5] eq "polar")

{

open OUT, ">$ARGV[0]"."$ARGV[5].txt";

while($ARGV[1] <= $ARGV[3])

{

open IN, "$ARGV[0]"."$ARGV[1]"."$ARGV[4]";

$i = 0;

while($line = <IN>)

{

chomp($line);

if($line =~ / Global net ELEC energy = \*/ && $i < 2)

{

$i++;

}

elsif($line =~ / Global net ELEC energy = \*/ && $i == 2)

{

@tmp = split(/\s+/, $line);

print OUT "@tmp[6] @tmp[7] ";

$line = @tmp[6];

@tmp = split(/E/, $line);

$tmp2 = @tmp[1];

@tmp = split(/\./, @tmp[0]);

if(substr($tmp2, 0, 1) eq "+")

{

$tmp2 = substr($tmp2, 1);

while($tmp2 > 0)

{

@tmp[0] .= substr(@tmp[1], 0, 1);

@tmp[1] = substr(@tmp[1], 1);

$tmp2--;

}

}

elsif(substr($tmp2, 0, 1) eq "-")

{

$tmp2 = substr($tmp2, 1);

if(@tmp[0] =~ /-\*/)

{

if($tmp2 < 0)

{

@tmp[1] = "substr(@tmp[0], 1)"."@tmp[1]";

@tmp[0] = "-0";

$tmp2++;

}

while($tmp2 < 0)

{

@tmp[1] = "0"."@tmp[1]";

$tmp2++;

}

}

else

{

if($tmp2 < 0)

{

@tmp[1] = "@tmp[0]"."@tmp[1]";

@tmp[0] = "0";

$tmp2++;

}

while($tmp2 < 0)

{

@tmp[1] = "0"."@tmp[1]";

$tmp2++;

}

}

}

else

{

print "\nWhat?\n";

exit;

}

print OUT "$tmp[0].$tmp[1] kJ/mol ";

$tmp2 = "$tmp[0].$tmp[1]";

$sum += $tmp2;

printf OUT "%.6f kcal/mol\n", ($tmp2 / 4.184);

}

}

close IN;

$ARGV[1]++;

}

printf OUT "\n\nAverage %.6f kcal/mol\n", (($sum / $num) / 4.184);

close OUT;

}

elsif($ARGV[5] eq "apolar")

{

open OUT, ">$ARGV[0]"."$ARGV[5].txt";

while($ARGV[1] <= $ARGV[3])

{

open IN, "$ARGV[0]"."$ARGV[1]"."$ARGV[4]";

while($line = <IN>)

{

chomp($line);

if($line =~ / Global net APOL energy = \*/)

{

@tmp = split(/\s+/, $line);

print OUT "@tmp[6] @tmp[7] ";

$line = @tmp[6];

@tmp = split(/E/, $line);

$tmp2 = @tmp[1];

@tmp = split(/\./, @tmp[0]);

if(substr($tmp2, 0, 1) eq "+")

{

$tmp2 = substr($tmp2, 1);

while($tmp2 > 0)

{

@tmp[0] .= substr(@tmp[1], 0, 1);

@tmp[1] = substr(@tmp[1], 1);

$tmp2--;

}

}

elsif(substr($tmp2, 0, 1) eq "-")

{

$tmp2 = substr($tmp2, 1);

if(@tmp[0] =~ /-\*/)

{

if($tmp2 < 0)

{

@tmp[1] = "substr(@tmp[0], 1)"."@tmp[1]";

@tmp[0] = "-0";

$tmp2++;

}

while($tmp2 < 0)

{

@tmp[1] = "0"."@tmp[1]";

$tmp2++;

}

}

else

{

if($tmp2 < 0)

{

@tmp[1] = "@tmp[0]"."@tmp[1]";

@tmp[0] = "0";

$tmp2++;

}

while($tmp2 < 0)

{

@tmp[1] = "0"."@tmp[1]";

$tmp2++;

}

}

}

else

{

print "\nWhat?\n";

exit;

}

print OUT "$tmp[0].$tmp[1] kJ/mol ";

$tmp2 = "$tmp[0].$tmp[1]";

$sum += $tmp2;

printf OUT "%.6f kcal/mol\n", ($tmp2 / 4.184);

}

}

close IN;

$ARGV[1]++;

}

printf OUT "\n\nAverage %.6f kcal/mol\n", (($sum / $num) / 4.184);

close OUT;

}

elsif($ARGV[5] eq "coulomb")

{

open OUT, ">$ARGV[0]"."$ARGV[5].txt";

while($ARGV[1] <= $ARGV[3])

{

open IN, "$ARGV[0]"."$ARGV[1]"."$ARGV[4]";

while($line = <IN>)

{

chomp($line);

if($line =~ /Total energy =\*/)

{

@tmp = split(/\s+/, $line);

print OUT "@tmp[3] @tmp[4] ";

$line = @tmp[3];

@tmp = split(/e/, $line);

$tmp2 = @tmp[1];

@tmp = split(/\./, @tmp[0]);

if(substr($tmp2, 0, 1) eq "+")

{

$tmp2 = substr($tmp2, 1);

while($tmp2 > 0)

{

@tmp[0] .= substr(@tmp[1], 0, 1);

@tmp[1] = substr(@tmp[1], 1);

$tmp2--;

}

}

elsif(substr($tmp2, 0, 1) eq "-")

{

$tmp2 = substr($tmp2, 1);

if(@tmp[0] =~ /-\*/)

{

if($tmp2 < 0)

{

@tmp[1] = "substr(@tmp[0], 1)"."@tmp[1]";

@tmp[0] = "-0";

$tmp2++;

}

while($tmp2 < 0)

{

@tmp[1] = "0"."@tmp[1]";

$tmp2++;

}

}

else

{

if($tmp2 < 0)

{

@tmp[1] = "@tmp[0]"."@tmp[1]";

@tmp[0] = "0";

$tmp2++;

}

while($tmp2 < 0)

{

@tmp[1] = "0"."@tmp[1]";

$tmp2++;

}

}

}

else

{

print "\nWhat?\n";

exit;

}

$tmp2 = "$tmp[0].$tmp[1]";

$tmp2 = $tmp2 / $ARGV[6];

print OUT "$tmp2 kJ/mol ";

$tmp2 = $tmp2 / 4.184;

printf OUT "%.6f kcal/mol\n", $tmp2;

$sum += $tmp2;

}

}

close IN;

$ARGV[1]++;

}

printf OUT "\n\nAverage %.6f kcal/mol\n", ($sum / $num);

close OUT;

}

else

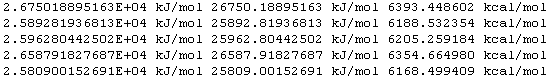
{

print "\nWhat??\n";

exit;

}

執行nohup sh run.01.sh > run.01.log & 在背景執行，最多執行兩個，可輸入jobs檢查是否已結束，跑完以後會出現一.polar.txt檔，該檔案會記錄所有能量，最後一欄為所需能量，.polar.txt格式如下



* apolar

執行vmd –e 01makePDBFromDCD.txt使每一個frame產生一個pdb檔，設定apolar.in檔以及vparam-amber-parm94.sans.20100422.dat

apolar.in只需更改parm flat，而vparam-amber-parm94.sans.20100422.dat需加入自己的藥物設定，其格式如下，欄位依序為藥物名稱(residue name)、atom name、charge、atom大小相關數值(位在parameter檔的nonbonded的第四欄，van der Waals radius in A) 、van der Waals well depth in kcal/mol

#M16

M16 C1 0.427 1.8600 0.359824000

M16 C2 0.309 1.8600 0.359824000

M16 C3 0.088 2.0200 0.359824000

M16 C4 -0.208 2.0100 0.359824000

M16 C5 0.264 2.2000 0.359824000

apolar.in

read

# mol pqr homo.Nit2.3ppa.H.dyna\_out.01.126.pqr

parm flat vparam-amber-parm94.sans.20100422.dat

end

#grid 0.2 0.2 0.2 + sdens 3600.0 + dpos 0.01 => 1.8G

APOLAR name solvated

grid 1.0 1.0 1.0

mol 1

srfm sacc

swin 0.3

srad 1.4

press 0.2394

gamma 0.0085

bconc 0.033428

sdens 4.0

dpos 0.000001

temp 300

calcenergy total

calcforce no

END

print apolEnergy solvated end

quit

將apolar.in、1001個frame pdb、run.01.sh、exeApbs01.pl、getApbsEnergy01.pl、vparam-amber-parm94.sans.20100422.dat檔案上傳至機房

run.01.sh

source apbsOn

fileName="equi\_equi\_01\_out\_chainazx."

date

perl exeApbs01.pl $fileName 0 1 1000 .pdb "pdb2pqr.py" "--ffout=AMBER --assign-only --ff=CHARMM" apbs apolar.in

#must use --ffout=AMBER at here for the resname and atomname in the vparam-amber-parm94.sans.\*.dat file

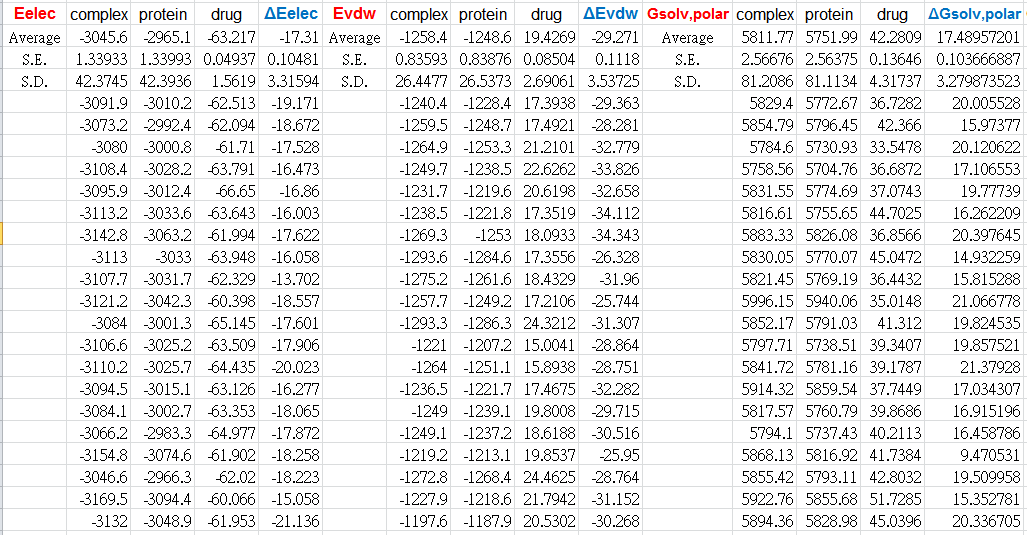
date

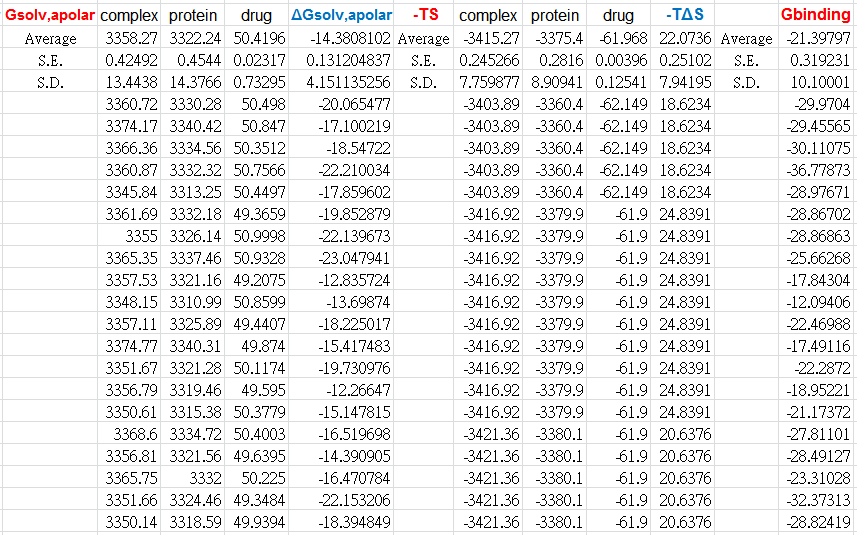
perl getApbsEnergy01.pl $fileName 0 1 1000 .log.txt apolar 2

date

執行nohup sh run.01.sh > run.01.log &，結束後會出現一.polar.txt檔，該檔案會記錄所有能量，最後一欄為所需能量

將所有的數值紀錄於excel，其格式如下，每一部分先進行complex-(protein+ligand)，在將最後的數值加起來，觀察其平均即可判斷其藥物與protein是傾向結合還是分開，若算出來的值為負數，則代表protein與ligand傾向於結成complex的狀態。



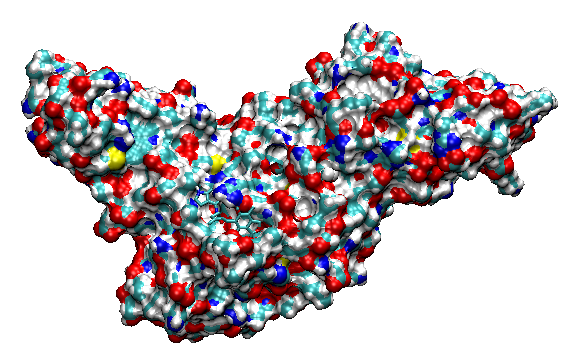


**Electrostatic Potential Surfaces**

有使用命令視窗及直接使用VMD操作介面兩種方式兩種方式，在操作之前需先到<https://sourceforge.net/projects/apbs/> 下載APBS並安裝

* **Method I**

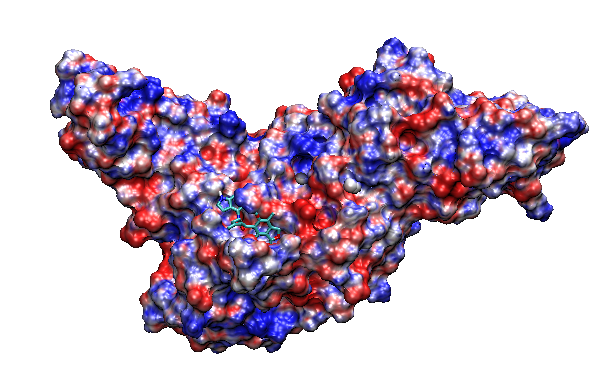
File=>New Molecule=>先讀取protein的psf在讀取pdb檔，Graphics=>Representations=>Drawing Method: Surf，將protein使用surface的表示方法(如下圖所示)



Extensions=>Analysis=>APBS Electrostatics，開啟APBS操作介面，上方的Edit=> Settings，位置分別選擇蛋白質位在的資料夾以及APBS套件位在的資料夾，勾選Set files only, do not run APBS=>OK，下方的Edit編輯要跑的分子，設定好以後按Run APBS，即會產生APBS需要input檔，會有apbs.in以及.pqr檔

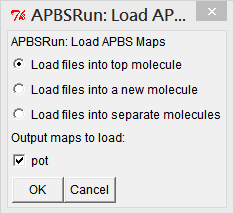
到該資料夾開啟命令視窗，輸入apbs apbs.in即會開始計算表面電荷，計算結束後會產生io.mc以及紀錄電荷的pot.dx檔

回到VMD，File=>New Molecule=>Load files for:選擇先前計算的蛋白質; Filename: pot.dx=>Load，將pot.dx檔讀進去後開啟Graphical Representations，Coloring Method: Volume，再到Trajectory設定Color Scale Data Range，範圍可設-10 10，若想要顏色強烈些可以設-4 4=>Set (結果如下圖)



* **Method II**

前面與方法一一樣，File=>New Molecule=>先讀取protein的psf在讀取pdb檔，Graphics=>Representations=>Drawing Method: Surf，將protein使用surface的表示方法，Extensions=>Analysis=>APBS Electrostatics，上方的Edit=> Settings，Working Directory要選擇該蛋白質的資料夾，設定好後按OK，接下來按Run APBS直接計算，計算結束以後會出現一視窗(如下圖)



選擇第一項按OK，開啟Graphical Representations，設定Coloring Method: Volume，再到Trajectory設定Color Scale Data Range即可完成

* **用Gaussian的charge建topology parameter**

開啟VMD讀取pdb檔，開TKConsole: paratool =>File=>Setup QM geometry optimization，更改Requested memory、Number of processors=>Write Gaussian input file，.com可改成.gjf，送上機房跑OPT

OPT完後把.log檔以及.chk檔抓下來，開啟VMD=> TKConsole: paratool=> File=> Import Gaussian optimized geometry (選取opt後的.log檔) => Hessian=> Setup QM single point calc. (Hessian, charges)=> 更改Requested memory、Number of processors=> Write Gaussian input file，.com可改成.gjf，送上機房跑第二次OPT

跑完以後須先load第一次的.log，再load第二次的.log，開啟VMD=> TKConsole: paratool=> File=> Import Gaussian optimized geometry (選取第一次opt後的.log檔) => Hessian=> Import Hessian/charges from single point calc.

接下來使用edit的功能來給予電荷以及VDW半徑，Edit=> Atom properties，先選取所有的原子改成NPA再點取<--按鍵，使charge使用NPA所算出來的電荷，再改Rname、Segid=> Submit change，Edit=> Auto assign type names，給予其type name，Edit=> Auto assign VDW parameters，給予其VDWeps及VDWrmin，再點選Write寫出topology以及parameter (#Br在給予type name、VDW parameter前要先手動更改element，因兩位數的元素名稱會出錯)

#需注意pdb的atom name需與topology檔一致

#需注意topology的總電荷是否為零，因小數點會無條件進位，故會有誤差

1. 開啟”psfgen”plugin [↑](#footnote-ref-1)
2. Topology: 讀取Topology檔案 [↑](#footnote-ref-2)
3. pdbalias: 修正結晶結構中可能會出現得不同的標示名稱 [↑](#footnote-ref-3)
4. 建立segment X，讀取pdb內容 [↑](#footnote-ref-4)
5. 讀取pdb中的座標 [↑](#footnote-ref-5)
6. 根據Top檔補齊缺少的座標 [↑](#footnote-ref-6)
7. writepdb/writepsf寫出檔案 [↑](#footnote-ref-7)
8. 退出VMD [↑](#footnote-ref-8)
9. Protein的.pdbqt檔 [↑](#footnote-ref-9)
10. Ligand的.pdbqt檔 [↑](#footnote-ref-10)
11. Center: Docking的中心點座標 [↑](#footnote-ref-11)
12. Size: Docking的範圍大小 [↑](#footnote-ref-12)
13. 輸出檔的名稱，可自行更改 [↑](#footnote-ref-13)
14. 記錄檔，可自行更改 [↑](#footnote-ref-14)
15. 設定在Docking過程中ligand的可能性 [↑](#footnote-ref-15)
16. 輸出結果的數量 [↑](#footnote-ref-16)
17. 輸出結果的的能量差距範圍 [↑](#footnote-ref-17)
18. noh atom的數量X3 [↑](#footnote-ref-18)