NWPesSe

User Manual

Version 1.0

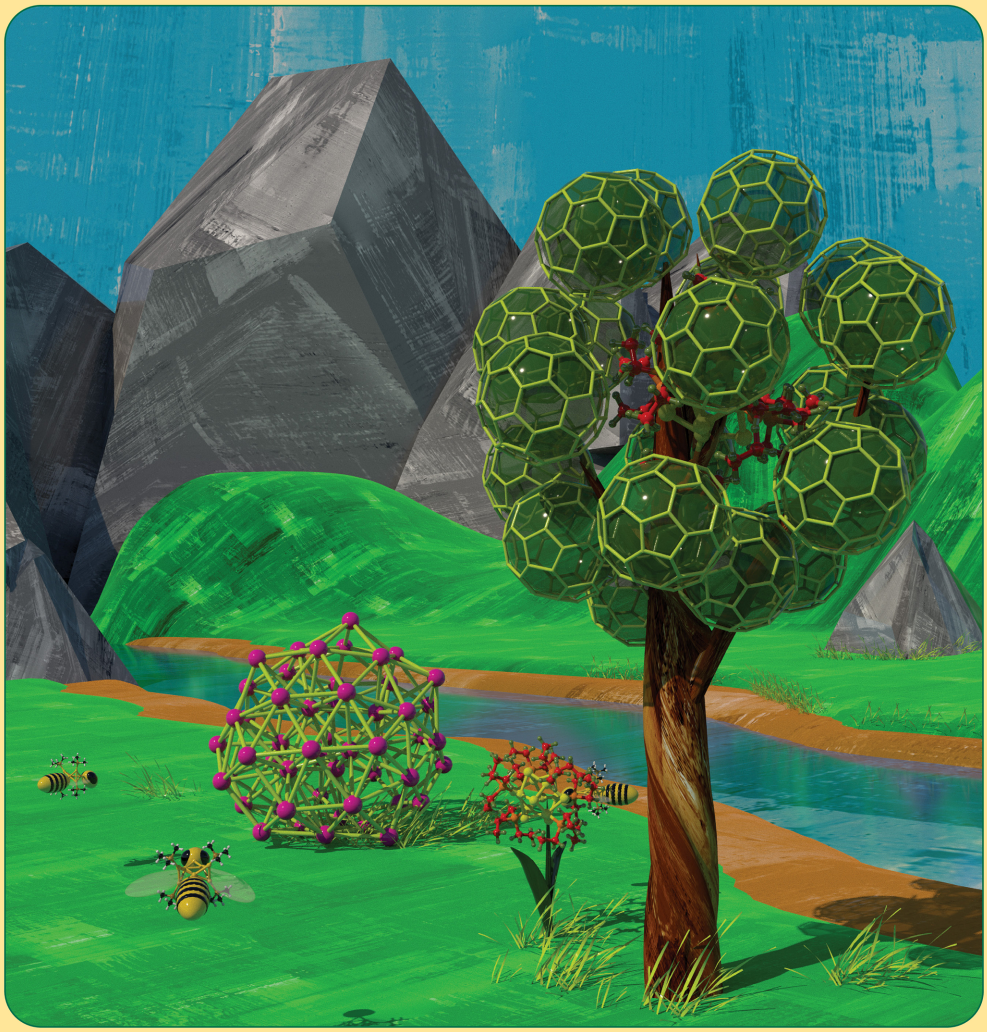
by Jun Zhang

more information:

https://store.pnnl.gov/content/

north-west-potential-energy-surface-search-engine-nwpesse

JCTC Journal of Chemical Theory and Computation
June 2020 Volume 16 Number 6 pubs.acs.org/JCTC



ACS Publications
Most Trusted. Most Cited. Most Read.
www.acs.org

Cover of NWPEsSe paper: *J. Chem. Theory Comput.* **2020**, *16*, 3947–3958.

Contents

Chapter 1

Introduction

1.1 What is NWPesSe?

Briefly, NWPesSe is short for "Northwest Potential Energy Surface Search Engine". It searches the **global** as well as the **local** minima of atomic and molecular clusters, with energies calculated using other computational chemical programs.

NWPesSe is a program developed by Dr. Jun Zhang and Dr. Vassiliki-Alexandra Glezakou at Pacific Northwest National Laboratory and can downloaded free of charge.

The latest version of NWPesSe and related resources can be obtained online:

`https://store.pnnl.gov/content/north-west-potential-energy-surface-search-engine-nwpesse`

NWPesSe uses the very efficient global optimization algorithm, based on the "the artificial bee colony algorithm" developed by Karaboga in 2005 (Karaboga, D. and Akay, B., 2009. A comparative study of artificial bee colony algorithm. Applied mathematics and computation, 214(1), pp.108-132), to perform the search. The algorithm has been modified and improved specifically for chemical problems. NWPesSe supports a large number of force fields by itself, and also provide interfaces for any other third-party computational chemistry programs, meaning that NWPesSe is able to generate reasonable initial structures of complex topology quickly by itself and also can generate optimized cluster structures using highly accurate *ab initio* quantum chemistry methods!

A first implementation of this method was adapted for the global optimization of small small atomic and rigid molecular clusters, you can try ABCluster which can be obtained from:

`http://www.zhjun-sci.com/software-abcluster-download.php`

1.2 Program Citation

For any published works using NWPesSe please include the following references

Zhang, J.; Glezakou, V.-A.; Rousseau, R.; Nguyen, M.-T. NWPEsSe: An Adaptive-Learning Global Optimization Algorithm for Nanosized Cluster Systems. J. Chem. Theory Comput. 2020, 16, 3947–3958.

Zhang, J.; Dolg, M. Global Optimization of Clusters of Rigid Molecules Using the Artificial Bee Colony Algorithm. Phys. Chem. Chem. Phys. 2016, 18, 3003–3010.

Zhang, J.; Dolg, M. NWPesSe: The Artificial Bee Colony Algorithm for Cluster Global Optimization. *Phys. Chem. Chem. Phys.* **2015**, *17*, 24173–24181.

1.3 Contacting the Authors

For question or comments contact: Vanda.Glezakou@pnnl.gov and Jun.Zhang@pnnl.gov .

1.4 Files of NWPesSe

When you download NWPesSe and unzip it, you can get several files:

- `lego` For the global optimization or fast initial structure guess of any clusters with third-party programs.
- `xyz2gaussian` A small program for NWPesSe using Gaussian.
- `gaussian2xyz` A small program for NWPesSe using Gaussian.
- `manual.pdf` The user manual of NWPesSe.
- `misc/dmol34nwpesse.sh` A script for NWPesSe using DMol3 (Courtesy of Prof. Dr. Lei Ma and Mr. Kai Wang, Tianjin International Center for Nanoparticles and Nanosystems (TICNN), http://ticnn.tju.edu.cn).
- `misc/mopac4nwpesse.sh` A script for NWPesSe using MOPAC.
- `misc/nwchem4nwpesse.sh` A script for NWPesSe using NWChem.
- `misc/orca4nwpesse.sh` A script for NWPesSe using ORCA.
- `misc/cp2k4nwpesse/*` Scripts for NWPesSe using CP2K.
- `misc/gromacs4nwpesse/*` Scripts for NWPesSe using Gromacs.
- `misc/lammps4nwpesse/*` Scripts for NWPesSe using Lammps.
- `testfiles` This folder contains all test input files.

1.5 Installation of NWPesSe

NWPesSe does not need to be installed. Please note that the current version of NWPesSe does not have a graphic user interface (GUI). One must use it in the command line. You can use NWPesSe with its absolute path every time. You can also set "PATH" variable to the directory where NWPesSe resides.

1.5.1 For Linux and Mac OS X Users

Assume you put NWPesSe (`atom-optimizer`, etc) in the directory `/home/you/bin/NWPesSe`, then open your `.bashrc`. At the end of the file, add the statement

```
export PATH=$PATH:/home/you/bin/NWPesSe
```

and run the command `source ~/.bashrc`.

1.5.2 For Linux and Mac OS X Users: Possible Issues

Sometimes you cannot run NWPesSe. If so, try the following ways.

- `/usr/lib64/libstdc++.so.6: version 'GLIBCXX_3.4.21' not found`: Please install or activate a GCC of higher version, like gcc 5.2.0. If you work on a computing cluster, maybe you can try `module load gcc/5.2.0`.
- `dyld: Library not loaded`: For Mac OS X users, you have to install `boost`, which can be done with two commands:

```
/usr/bin/ruby -e "\$(curl -fsSL https://raw.githubusercontent.com/Homebrew/
    install/master/install)"
brew install boost
```

Chapter 2

Optimization with Third-party Programs: `lego`

2.1 Introduction

`lego` has now become a very powerful cluster building tool. One can generate much more complex structures than before. Thus, it is not only a global optimization program, but can also be used to generate initial guess for molecular dynamics or quantum chemistry. In this chapter we will discuss it in detail.

2.2 `lego` by Itself: Generate Complex Clusters

You can find all input and output files in `testfiles/lego/1-9-mol`.

In this section, we will describe how to use `lego` to generate complex clusters. No third-party programs are needed. Therefore, you can use `lego` to generate initial structures for molecular dynamics or quantum chemistry calculations.

2.2.1 Preparation

Step 1: In the NWPEsSe distribution, go to the directory `testfiles/lego`. Create a file called `mol.inp`. For Linux and Mac OS X users,

```
mol              # Result file name
mol.cluster      # Cluster file name
5                # Maximum number of calculations
>>>>

>>>>
cp $inp$ $out$
>>>>
```

The meaning of each line is:

- Line 1: The name of the directory that stores the found local minima (LM). After optimization, all the found LMs will be stored in `mol-LM`.
- Line 2: The name of the file that contains the cluster components and geometry. This is very similar to that for `rigidmol-optimizer`.
- Line 3: The number of calculations you want to do. Here it means `lego` will do 5 calculations.

- The lines between two `>>>>` are the structure features, i.e., what shape of the cluster is. They will be explained in detail in Subsection 2.2.4.
- The lines between the next two `>>>>` are the commands to call third-party programs to calculate the energy. This is explained in Section **??** and is completely identical to that for `isomer`.

You can see that, the command `copy $inp$ $out$` or `cp $inp$ $out$` simply send the generated structure back to NWPEsSe without being optimized by other programs.

Step 2: Prepare some molecules in XYZ format.

Water `h2o.xyz`:

```
3
water
 O                  0.00000000     0.00000000    -0.11081188
 H                  0.00000000    -0.78397589     0.44324751
 H                 -0.00000000     0.78397589     0.44324751
```

Na^+ `na.xyz` and SO_4^{2+} `so4.xyz`:

```
1
Na+
Na  0.  0.  0.
```

```
5
SO4(2-)
 S                  0.00000000     0.00000000     0.00000000
 O                  0.00000000     0.00000000     1.67000004
 O                 -0.00000000    -1.57449114    -0.55666668
 O                  1.36354933     0.78724557    -0.55666668
 O                 -1.36354933     0.78724557    -0.55666668
```

Phenylacetylene C_6H_5CCH `c6h5cch.xyz`:

```
14
C6H5CCH
 C                  0.00000000    -1.20805470    -1.53350228
 C                  0.00000000    -1.20800413    -0.13834242
 C                  0.00000000    -0.00000000     0.55915207
 C                 -0.00000000     1.20800413    -0.13834242
 C                 -0.00000000     1.20805470    -1.53350228
 C                  0.00000000    -0.00000000    -2.23092797
 H                  0.00000000    -2.16039172    -2.08322677
 H                  0.00000000    -2.16049784     0.41120038
 H                 -0.00000000     2.16049784     0.41120038
 H                 -0.00000000     2.16039172    -2.08322677
 H                  0.00000000    -0.00000000    -3.33053195
 C                  0.00000000    -0.00000000     2.09915204
 C                  0.00000000    -0.00000000     3.30035204
 H                  0.00000000    -0.00000000     4.37035204
```

An icosahedral cluster Au_{55} `au55.xyz`:

```
55
Au55, Ih
```

Au	0.00000000	0.00000000	-0.00000000
Au	0.00000000	0.00000000	2.78370787
Au	0.00000099	-2.48982349	1.24491305
Au	-2.36796255	-0.76939871	1.24491305
Au	2.36796255	-0.76939871	1.24491305
Au	-1.46348073	2.01431010	1.24491305
Au	1.46348233	2.01430893	1.24491305
Au	-1.46348134	-2.01431199	-1.24490928
Au	1.46348134	-2.01431199	-1.24490928
Au	-2.36796415	0.76939988	-1.24490928
Au	2.36796515	0.76939683	-1.24490928
Au	0.00000160	2.48982538	-1.24490928
Au	-0.00000000	0.00000000	-2.78370787
Au	0.00000000	0.00000000	5.60588695
Au	0.00000200	-5.01405667	2.50703096
Au	-4.76865065	-1.54943062	2.50703096
Au	4.76865065	-1.54943062	2.50703096
Au	-2.94718695	4.05645823	2.50703096
Au	2.94719018	4.05645588	2.50703096
Au	-2.94718818	-4.05646203	-2.50702336
Au	-4.76865388	1.54943297	-2.50702336
Au	4.76865588	1.54942682	-2.50702336
Au	0.00000323	5.01406047	-2.50702336
Au	-0.00000000	0.00000000	-5.60588695
Au	0.00000000	-2.60015464	4.20713602
Au	-2.47289401	-0.80349197	4.20713602
Au	2.47289401	-0.80349197	4.20713602
Au	-1.52833255	2.10356929	4.20713602
Au	1.52833255	2.10356929	4.20713602
Au	-2.47288912	-3.40364611	2.60015580
Au	2.47288912	-3.40364611	2.60015580
Au	-1.52832953	-4.70372262	0.00000718
Au	1.52832953	-4.70372262	0.00000718
Au	-4.00122458	1.30007635	2.60015404
Au	-4.00122606	-2.90705825	0.00000028
Au	-4.94578585	0.00000153	0.00000142
Au	4.00122458	1.30007635	2.60015404
Au	4.00122606	-2.90705825	0.00000028
Au	4.94578585	0.00000153	0.00000142
Au	0.00000164	4.20713581	2.60015498
Au	-4.00122542	2.90705913	0.00000542
Au	-1.52833164	4.70372193	0.00000133
Au	4.00122542	2.90705913	0.00000542
Au	1.52833164	4.70372193	0.00000133
Au	-0.00000260	-4.20713750	-2.60015224
Au	-4.00122857	-1.30007547	-2.60014834
Au	-1.52833303	-2.10356685	-4.20713707
Au	4.00122857	-1.30007547	-2.60014834
Au	1.52833303	-2.10356685	-4.20713707
Au	-2.47289130	3.40364805	-2.60015119
Au	-2.47289519	0.80349588	-4.20713458

```
 Au                  2.47289130     3.40364805    -2.60015119
 Au                  2.47289519     0.80349588    -4.20713458
 Au                 -0.00000000     2.60015758    -4.20713421
 Au                  2.94718818    -4.05646203    -2.50702336
```

Step 3: Prepare a file `mol.cluster` for later use.

2.2.2 Example 1: $Na_4(SO_4)_2(H_2O)_{300}$

We consider $(Na_4(SO_4)_2(H_2O)_{300}$. Of course, it can be perfectly done with `rigidmol-optimizer`. But here we just want to demonstrate how to use `lego`.

Step 1: Modify `mol.cluster`. This is exactly the same as `rigidmol-optimizer`.

```
3
h2o.xyz    300
na.xyz     4
so4.xyz    2
```

Step 2: Modify `mol.inp`. If you consider an isolated cluster, it should be like a sphere. But assume for some reasons, you **want** the cluster be in rectangle shape (say, for molecular dynamics simulations). This can be done by using the following input:

```
mol              # Result file name
mol.cluster      # Cluster file name
5                # Maximal number of calculations
>>>>
inbox 0. 0. 0. 8. 8. 20.
inbox 0. 0. 0. 8. 8. 20.
inbox 1. 1. 1. 7. 7. 17.
>>>>
cp $inp$ $out$
>>>>
```

Here, the first line `inbox 0. 0. 0. 8. 8. 20.` means that the first component of the cluster, i.e., 300 H_2O, will be distributed randomly in a box defined by (0,0,0) and (8,8,20) (Figure 2.2.1); the second line `inbox 0. 0. 0. 8. 8. 20.` means that the second component of the cluster, i.e., 4 Na^+, will be distributed randomly in a box defined by (0,0,0) and (8,8,20); the third line `inbox 1. 1. 1. 7. 7. 17.` means that the third component of the cluster, i.e., 2 ${SO_4}^{2-}$, will be distributed randomly in a box defined by (1,1,1) and (7,7,17). Note that for ${SO_4}^{2-}$, we let them distribute in a smaller box.

Step 3: Run the program:

```
lego mol.inp > mol.out
```

Step 4: All the 5 structures have the same energy `0.00000` because no programs are used to calculate them. Go to `mol-LM` and you will find 5 different structures. Pick the one you need. For example, `mol-LM/3.xyz` is shown in Figure 2.2.2. Note that the shape is like a rectangle. That is because the box is too small so water molecules overflow.

2.2.3 Example 2: $(H_2O)_{1000}(C_6H_5CCH)_{150}$

Now we build a solvent box with 1000 H_2O and 150 C_6H_5CCH. We only need to modify `mol.cluster` and structural feature part of `mol.inp`, like:

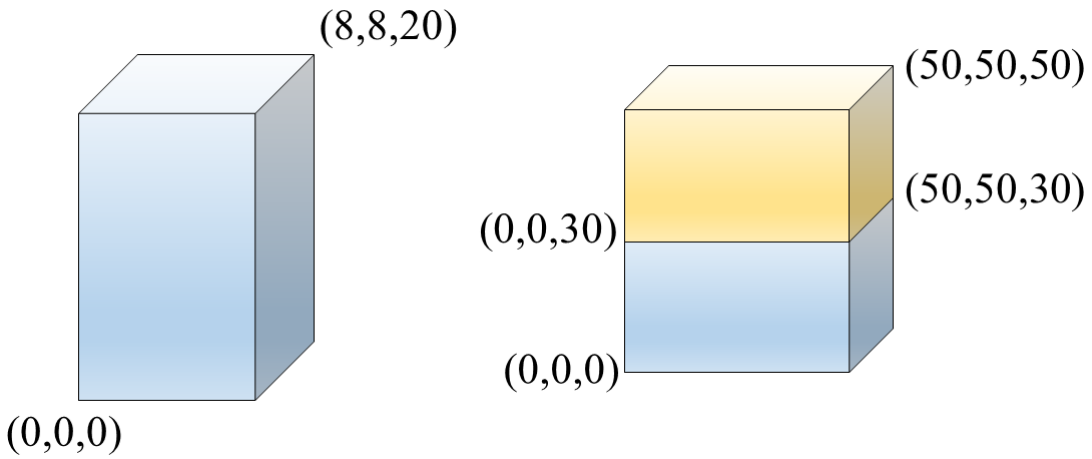


Figure 2.2.1: Definitions of `inbox`.

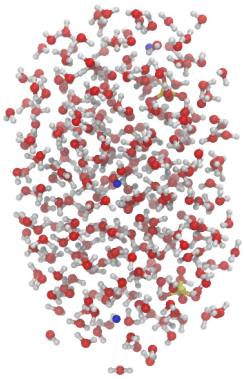


Figure 2.2.2: One example of $Na_4(SO_4)_2(H_2O)_{300}$.

```
mol.cluster:
2
h2o.xyz       1000
c6h5cch.xyz   150
mol.inp
inbox 0. 0. 0. 50. 50. 50.
inbox 0. 0. 0. 50. 50. 50.
```

These input files will place 1000 H_2O and 150 C_6H_5CCH in a box $50 \times 50 \times 50$ (Figure 2.2.1).

Run the program with `lego mol.inp > mol.out`. You will find structures in `mol-LM`. For example, `mol-LM/3.xyz` is shown in Figure 2.2.3.

However, we know that C_6H_5CCH is not soluble in H_2O, therefore it is more suitable to build a two-phase structure. We can use:

```
mol.cluster:
2
h2o.xyz       1000
c6h5cch.xyz   150
mol.inp
```

```
inbox 0. 0. 0. 50. 50. 30.
inbox 0. 0. 30. 50. 50. 50.
```

These input files will place 1000 H_2O and 150 C_6H_5CCH in two separated but neighboring boxes (Figure 2.2.1). Use `lego` to run this optimization. One result can be found in 2.2.3.

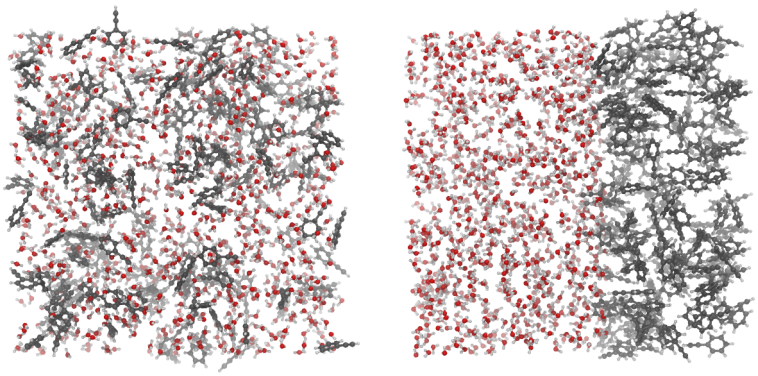


Figure 2.2.3: Cluster structures of $(H_2O)_{1000}(C_6H_5CCH)_{150}$.

A tip: **If you want to use lego to generate boxes for molecular dynamics, the number of molecules and box size should be compatible with the density of solvents.**

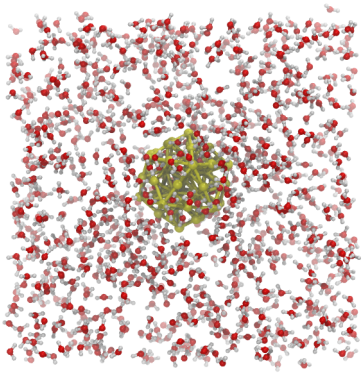
2.2.4 Example 3-9: All Structural Features

Now we introduce all structrual features that `lego` supports besides `inbox`. You can combine them to get very complex cluters.

1. `pos x y z alpha beta gamma` Put the molecule at a point (x,y,z) in an orientation (alpha,beta,gamma).

   Example 3: put Au_{55} at (20,20,20) with no rotation (0,0,0) relative to its original orientation (defined in `au55.xyz`), and put 1000 H_2O in the box defined by (0,0,0) and (40,40,40).

```
mol.cluster:
2
au55.xyz      1
h2o.xyz       1000
mol.inp
pos    20. 20. 20. 0. 0. 0.
inbox   0. 0. 0. 40. 40. 40.
```



2. `inbox x1 y1 z1 x2 y2 z2` Put molecules inside the box defined by (x1,y1,z1) and (x2,y2,z2).

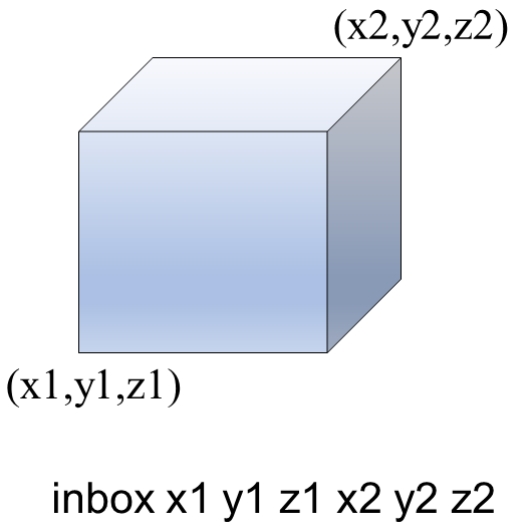


Figure 2.2.4: Definition of `inbox x1 y1 z1 x2 y2 z2`.

Example 4: See Subsection 2.2.2 and 2.2.3.

3. `outbox x1 y1 z1 x2 y2 z2 x3 y3 z3 x4 y4 z4` Put molecules outside the box defined by (x1,y1,z1) and (x2,y2,z2) but inside the box defined by (x3,y3,z3) and (x4,y4,z4).

   Example: put Au_{55} at (20,20,20) with no rotation (0,0,0) relative to its original orientation (defined in `au55.xyz`), and put 1000 H_2O in the box defined by (0,0,0) and (40,40,40), but outside the box defined by (14,14,14) and (26,26,26). Note that the size of Au_{55} is about $12 \times 12 \times 12$, thus (14,14,14) and (26,26,26) define the box enclosing Au_{55}.

```
mol.cluster:
2
au55.xyz      1
h2o.xyz       1000
mol.inp
pos   20. 20. 20. 0. 0. 0.
outbox 14. 14. 14. 26. 26. 26. 0. 0. 0. 40. 40. 40.
```

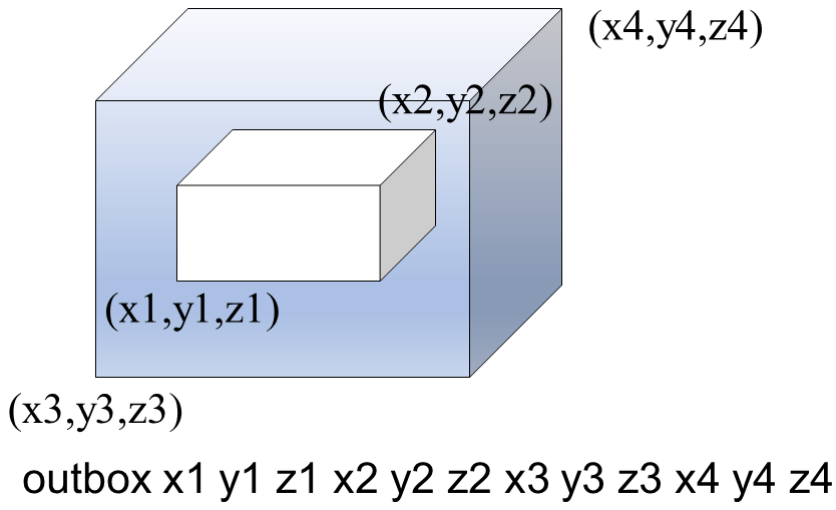
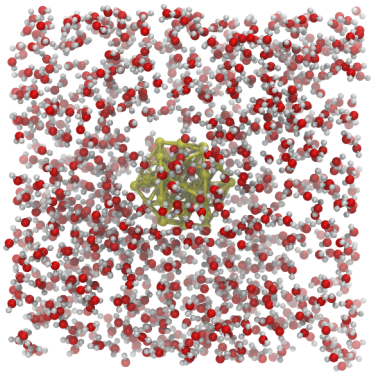


Figure 2.2.5: Definition of `outbox x1 y1 z1 x2 y2 z2 x3 y3 z3 x4 y4 z4`.



You can see that unlike the first example, there is no water existing inside the gold cluster. `outbox` can be very useful when, for example, you want to soak a large cage molecule (like fullerene) in a solvent box, without any solvent molecules inside the cage.

4. `insphere x y z ra rb rc` Put molecules inside a sphere centered at (x,y,z) with three radii ra, rb. and rc.

   Example 5: prepare a prolate sphere containing 1000 H_2O, and put a $SO_4^{(}{}_2-)$ near the surface of the water droplet.

```
mol.cluster:
2
h2o.xyz     1000
so4.xyz     1
mol.inp
insphere 0 0 0 20 25 10
pos 5 9 6 0 0 0
```

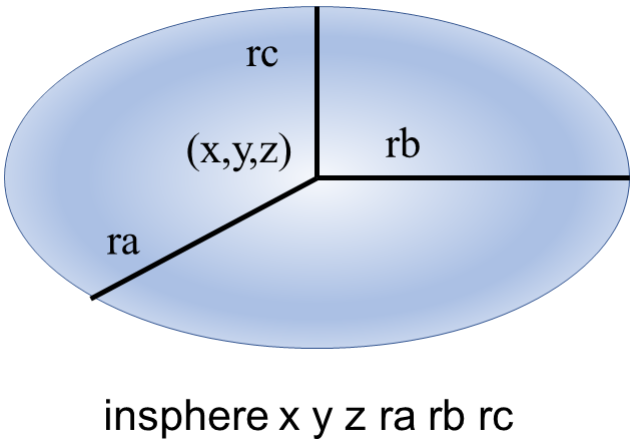
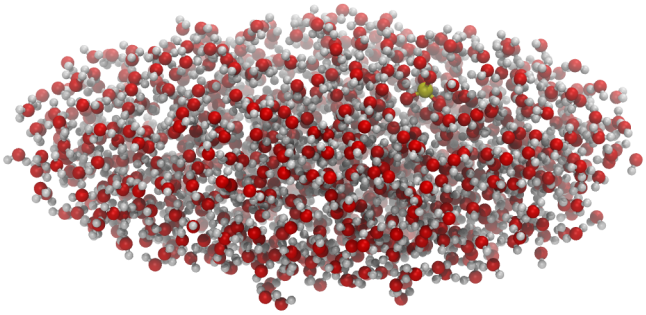


Figure 2.2.6: Definition of `insphere x y z ra rb rc`.



5. `onsurface n1 n2 n3` Put molecules randomly on surfaces of the n1-th component. For the molecules, the n2-th atom binds to the surface, the line defined by the n2-th and n3-th atom being othorgonal to the surface.

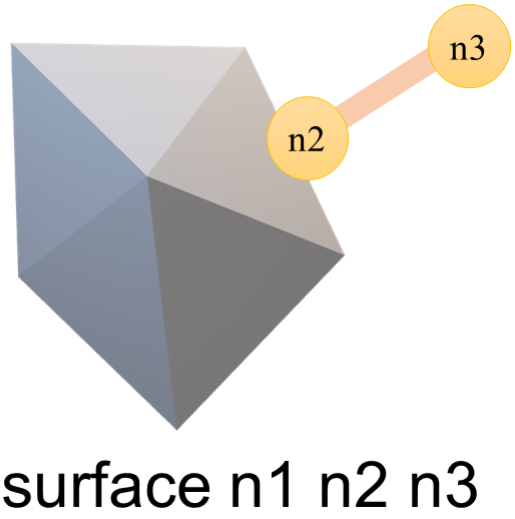
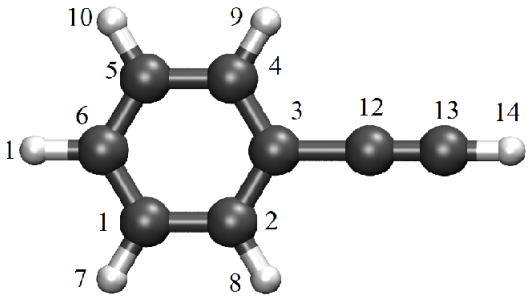
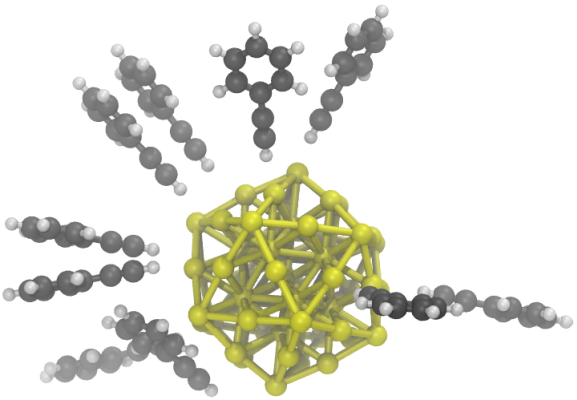


Figure 2.2.7: Definition of `onsurface n1 n2 n3`.

Example 6: 10 C_6H_6CCH are absorbed randomly on Au_{55}, with $\equiv C-H$ pointing toward Au_{55}.

```
mol.cluster:
2
au55.xyz        1
c6h5cch.xyz     10
mol.inp
pos 0 0 0 0 0 0
onsurface 1 14 13
```





6. `micelle n1 n2 x y z ra rb rc` Put molecules **uniformly** on a sphere surface centered at (x,y,z) with three radii ra, rb. and rc. For the molecules, the n1-th atom is on the sphere surface, the line defined by the n1-th and n2-th atom being othorgonal to the sphere surface.

   This option is very powerful to define a series of elegant structures. You will see this in the current and following sections.

   Example 7: Put 52 Na on a elliptic surface centered at (1,1,1) with radii 9,4,3.

```
mol.cluster:
1
na.xyz 52
mol.inp
micelle 1 1 0.5 0.5 0.5 9 4 3
```

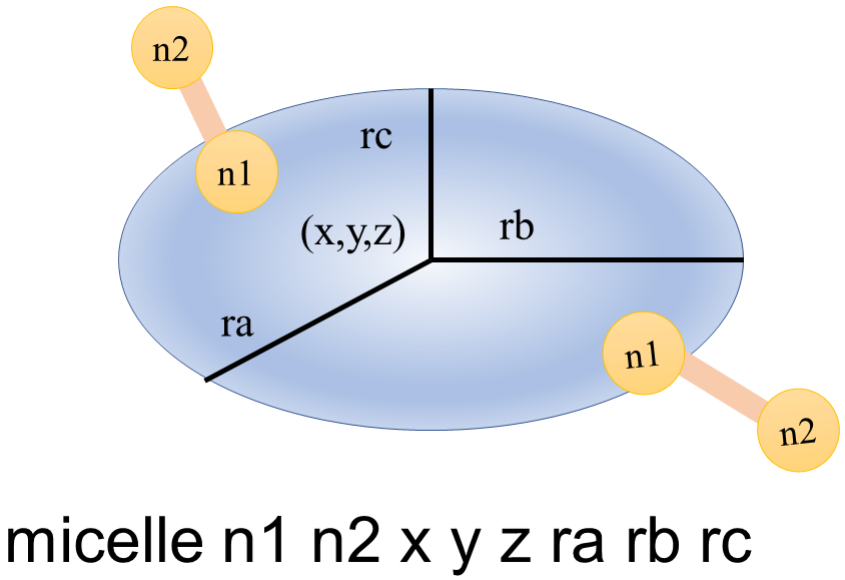
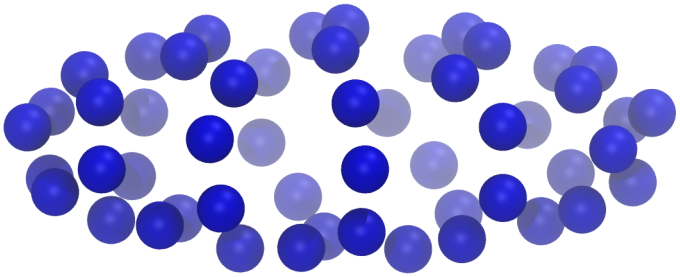


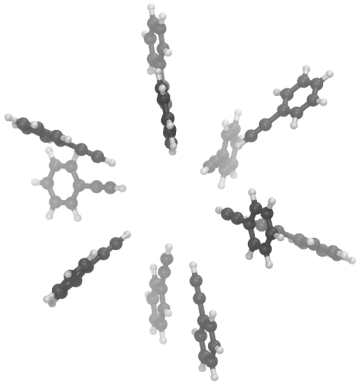
Figure 2.2.8: Definition of `micelle n1 n2 x y z ra rb rc`.



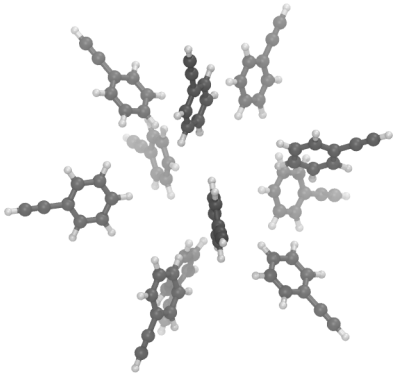
Note that there is only 1 atom in Na, therefore we simply write `1 1`.

Example 8: Put 11 C_6H_5CCH on a sphere surface centered at (0,0,0) with radius 5.

```
mol.cluster:
1
c6h5cch.xyz 11
mol.inp
micelle 14 13 0. 0. 0. 5. 5. 5.
```



What if we write `micelle 11 6 0. 0. 0. 5. 5. 5.`? Yes, the molecules change their directions.



7. `micelle n1 n2 n3 n4 x y z ra rb rc` Put molecules evenly on a sphere surface centered at (x,y,z) with three radii ra, rb, and rc. For the molecules, the center of the n1-th and n2-th atom is on the sphere surface, the line defined by the center of the n1-th and n2-th atom, and the center of the n3-th and n4-th atom, being othorgonal to the sphere surface.

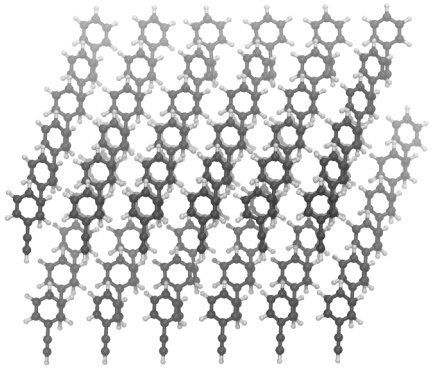
   `micelle n1 n2 n3 n4 n5 n6 x y z ra rb rc` Put molecules evenly on a sphere surface centered at (x,y,z) with three radii ra, rb. and rc. For the molecules, the center of the n1-th, n2-th and n3-th atom is on the sphere surface, the line defined by the center of the n1-th, n2-th and n3-th atom, and the center of the n4-th, n5-th and n6-th atom, being othorgonal to the sphere surface.

   This is useful when some cone-shape molecules are involved. See Section 2.4.1.

8. `layer n1 n2 n3 x1 y1 z1 x2 y2 z2 x3 y3 z3 x4 y4 z4` Align the molecules in a plane region defined by (x2,y2,z2), (x3,y3,z3), and (x4,y4,z4). For the molecules, For the molecules, the n1-th atom is on the plane, the line defined by the n1-th and n2-th atom being parallel to (x1,y1,z1). When n3 is 0, the molecules are distributed in the plane uniformly, otherwise randomly.

   Example 9: Construct two layers of 72 C_6H_5CCH.

```
mol.cluster:
2
c6h5cch.xyz 36
c6h5cch.xyz 36
mol.inp
layer 14 13 0 0. 1. 1. 0. 0. 0. 35. 0. 0. 0. 35. 0.
layer 13 14 0 0. 1. 1. 0. 0. 8. 35. 0. 8. 0. 35. 8.
```



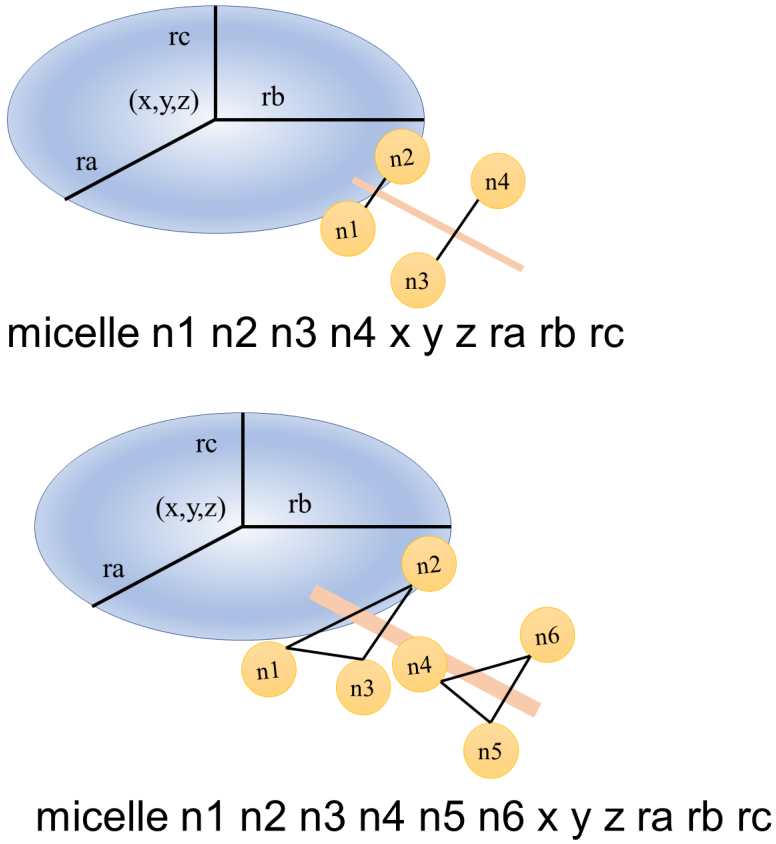


Figure 2.2.9: Definitions of `micelle n1 n2 n3 n4 x y z ra rb rc` and `micelle n1 n2 n3 n4 n5 n6 x y z ra rb rc`.

Using these structural features, you can build clusters of any complex shapes for quantum chemistry or molecular dynamics. Now we will show how to combine `lego` with other programs to do global optimizations for complex systems of different chemical nature.

2.3 `lego` with Gaussian

Gaussian is one of the most popular quantum chemistry program. It is fast and reliable in geometriy optimization. For small atomic and molecular clusters, it is a good choice.

2.3.1 Example 10: $(IO_3^-)(H_2O)_4$

You can find all input and output files in `testfiles/lego/10-h2o4io3`.

We are going to search the global minimum of $(IO_3^-)(H_2O)_4$ at B3LYP-D3/SVP level of theory. It relates to some studies of atmospheric nucleation processes (*J. Phys. Chem. Lett.* **2019**, *10*, 1935).

First prepare 2 files containing the coordinates of IO_3^-, and H_2O.

`io3.xyz`:

```
4
```

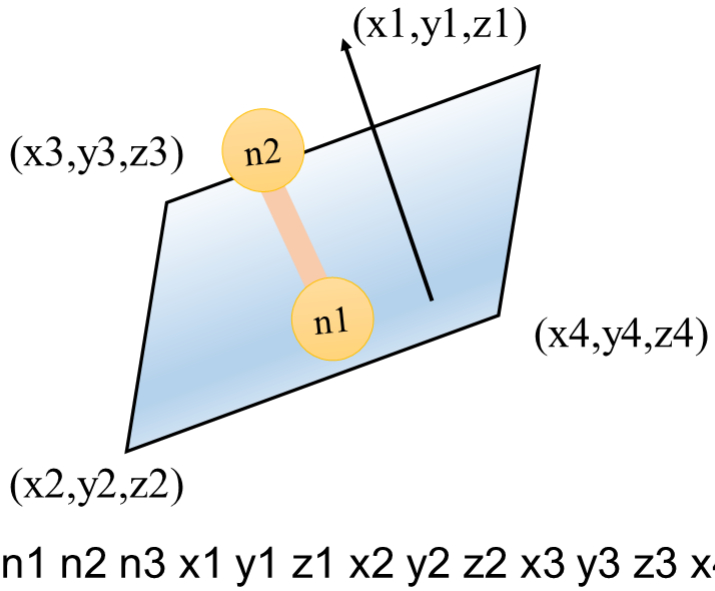


Figure 2.2.10: Definition of `layer n1 n2 n3 x1 y1 z1 x2 y2 z2 x3 y3 z3 x4 y4 z4`.

```
IO3(-)
 I                    0.00000000     0.00000000     0.20675328
 O                   -1.62482819    -0.93809499    -0.45658026
 O                   -0.00000000     1.87618998    -0.45658026
 O                    1.62482819    -0.93809499    -0.45658026
```

`h2o.xyz`:

```
3
H2O
 O                    0.00000000     0.00000000    -0.11081188
 H                    0.00000000    -0.78397589     0.44324751
 H                   -0.00000000     0.78397589     0.44324751
```

Prepare the cluster file `h2o4io3.cluster`:

```
2
h2o.xyz    4
io3.xyz    1
```

We will put the molecules randomly together in a box of size $7 \times 7 \times 7$, and do 50 calculations to search the global minimum. Thus the NWPEsSe input file `h2o4io3.inp` is:

```
h2o4io3i2o5              # Result file name
h2o4io3i2o5.cluster      # Cluster file name
50                       # Maximal number of calculations
>>>>
inbox 0. 0. 0. 7. 7. 7.
inbox 0. 0. 0. 7. 7. 7.
>>>>
./xyz2gaussian optfile $inp$ > $xxx$.gjf
```

```
g09 < $xxx$.gjf > $xxx$.log 2>/dev/null
./gaussian2xyz $xxx$.log > $out$
>>>>
```

The charge of the system is −1, thus `optfile`:

```
%nproc=16
%mem=24GB
#N B3LYP/SVP Empiricaldispersion(GD3) SCF(XQC) Geom(NoCrowd)
    Opt

opt

-1 1
>>>>


>>>>
```

With these files, run the global optimization:

```
nohup lego h2o4io3.inp > h2o4io3.out &
```

After optimization, in the output file `h2o4io3.out`, you can find:

```
Reordered from low to high energy:
================================================================

    #           Energy
================================================================

   41   -7450.79461219
   32   -7450.79277832
   21   -7450.79196832
   38   -7450.79070065
   16   -7450.79021089
```

The structure `41` is the global minimum. You can check it from `h2o4io3-LM/41.xyz`, shown in Figure 2.3.1.

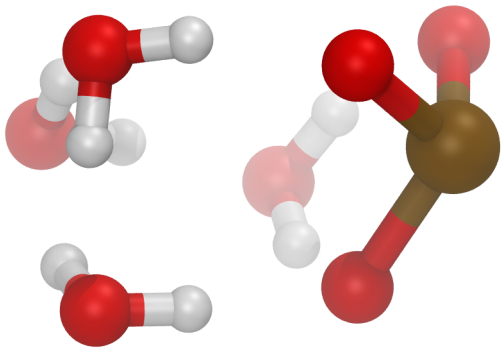


Figure 2.3.1: The global minimum of $(IO_3^-)(H_2O)_4$.

2.3.2 Example 11: $((NH_3)_4^-)$

You can find all input and output files in `testfiles/lego/11-enh34`.

Now we search the global minimum of $((NH_3)_4^-)$, which is a small model of ammonia-solvated electron.

Prepare the geometry of NH_3 `nh3.xyz`:

```
 4
NH3
 N                  -0.00000000    -0.00000000    -0.10000001
 H                   0.00000000    -0.94280900     0.23333324
 H                  -0.81649655     0.47140450     0.23333324
 H                   0.81649655     0.47140450     0.23333324
```

The cluster file `enh34.cluster`:

```
1
nh3.xyz    4
```

The input file `enh34.inp`:

```
enh34              # Result file name
enh34.cluster      # Cluster file name
40                 # Maximal number of calculations
>>>>
inbox 0. 0. 0. 4. 4. 4.
>>>>
./xyz2gaussian optfile $inp$ > $xxx$.gjf
g09 < $xxx$.gjf > $xxx$.log 2>/dev/null
./gaussian2xyz $xxx$.log > $out$
>>>>
```

Finally, edit `optfile`:

```
%nproc=16
%mem=24GB
#N B3LYP/6-31+g(d) Empiricaldispersion(GD3) SCF(XQC) Geom(
   NoCrowd) Opt

opt

-1 2
>>>>


>>>>
```

Note that for this solvated electron system, the diffuse basis set `6-31+g(d)` is applied. Also, the spin multiplicity is set as `2`.

With these files, run the global optimization:

```
nohup lego enh34.inp > enh34.out &
```

The global minimum is `enh34-LM/39.xyz`, shown in Figure 2.3.2.

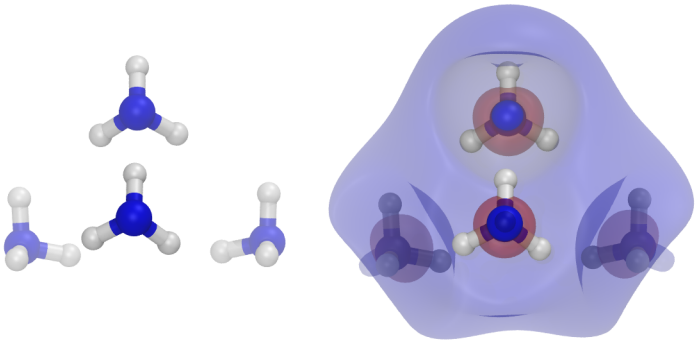


Figure 2.3.2: The global minimum of $(NH_3)_4^-$, and the isosurface of the single occupied molecular orbital (isovalue = ± 0.02).

2.3.3 Theoretical Background

For the mechanism how NWPEsSe works with Gaussian, please refer to Section **??**.

For different systems, You can change `optfile` to get the best performance. Below are some tips.

- Always add `SCF(XQC) Geom(NoCrowd)` to get stable convergences.
- Usually double-zeta basis sets (6-31g(d), cc-pVDZ, SVP) are enough for global optimization. Triple basis sets will waste your computation time. Except for a few cases, diffuse basis sets are also unnecessary.
- For clsuters containing no transition metal atoms, B3LYP-D3 is always a good choice; otherwise, pure functionals like BLYP with dispersion corrections may be used.
- For open-shell systems, like $(NH_3)_4^-$ or metal atomic clusters, set correct spin multiplicity in `optfile`. You may even have to consider spin symmetry broken.
- If you want to freeze some degrees of freedom, use `Opt(ModRedundant)` and add coordinates to be frozen between the two `>>>>`.
- If you want to use some special basis sets (like pseudopotentials), add basis set information in `optfile` between the two `>>>>`. See Subsection **??**.

2.4 `lego` with xTB

xTB is developed by Prof. Stefan Grimme and his coworkers. This is a very accurate and efficient semi-empirical method, and is **highly recommended** for first exploration of molecular clusters. The program can be obtained by contacting the authors (mailto: xtb@thch.uni-bonn.de).

2.4.1 Example 12: $Au_8(PPh_3)_7^{2+}$

You can find all input and output files in `testfiles/lego/12-au8pph37`.

$Au_8(PPh_3)_7^{2+}$ is a typical case of ligand-protected metal clusters. Using NWPEsSe+xTB, you can obtain its global minimum in a very simple but reliable way.

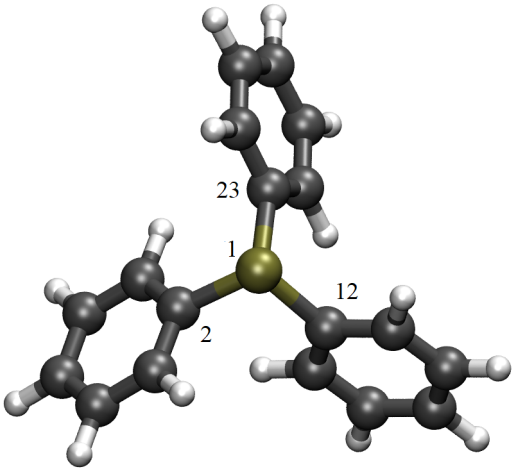
Prepare the structures of Au and PPh_3.

`au.xyz`:

```
1
Au
Au  0.  0.  0.
```

pph3.xyz:

```
 34
 PPh3
 P                  -0.03267416    1.25444481    0.01586741
 C                   1.63658711    0.51777952   -0.20249064
 C                   2.11728961   -0.51166267    0.62864297
 C                   3.35184041   -1.10947100    0.34809282
 C                   4.10821561   -0.69414585   -0.75395341
 C                   3.63530020    0.34051056   -1.57197305
 C                   2.40795850    0.94805830   -1.29666426
 H                   5.07175713   -1.17346984   -0.97083452
 H                   1.50773709   -0.86424610    1.47175488
 H                   2.05116452    1.76406952   -1.94166102
 H                   4.23354275    0.68320377   -2.42887611
 C                  -1.01544355    0.47507538   -1.33319588
 C                  -2.21940469    1.07011104   -1.74867816
 C                  -3.02846596    0.43629819   -2.69607994
 C                  -2.63622633   -0.78935952   -3.24540929
 C                  -1.42903176   -1.37711749   -2.85016511
 C                  -0.61623556   -0.75189941   -1.89960309
 H                   0.33989220   -1.20508238   -1.60281598
 H                  -3.27166241   -1.28124289   -3.99569753
 H                  -1.11558128   -2.34174977   -3.27817968
 H                  -2.52732156    2.03612934   -1.31911157
 H                   3.71666214   -1.92176100    0.99789092
 C                  -0.66142082    0.46509284    1.55752006
 C                  -0.39744126    1.10286726    2.78269286
 C                  -0.76708153    0.49574035    3.98770397
 C                  -1.41331451   -0.74647719    3.97484028
 C                  -1.69557473   -1.37249081    2.75456160
 C                  -1.32638551   -0.77586757    1.54359431
 H                  -1.69985780   -1.22476738    4.92127414
 H                  -2.20016600   -2.35118707    2.73885049
 H                  -1.52959169   -1.27689951    0.58452673
 H                  -3.96763868    0.91740951   -3.01466482
 H                  -0.53008388    1.00001913    4.93837267
 H                   0.10426851    2.08465110    2.78454401
```



Prepare the cluster file `au8pph37.cluster`:

```
2
au.xyz     8
pph3.xyz   7
```

Now we are going to prepare `au8pph37.inp`. Note that, it will be very inefficient to use `inbox` to put the molecules in a box. We know that from chemical intuition, the cluster will have a gold core, and PPh_3 binds on the surfaces of the gold core with its phosphorous atom. Thus, we put gold atoms in a sphere, then put ligands on the sphere surface.

```
au8pph37          # Result file name
au8pph37.cluster  # Cluster file name
200               # Maximal number of calculations
>>>>
insphere 0 0 0 5 5 5
micelle 1 1 1 2 12 23 0 0 0 6 6 6
>>>>
cp $inp$ $xxx$.xyz
../xtb/bin/xtb $xxx$.xyz -gfn 1 -chrg +2 -o > $xxx$.out
energy=‘awk ’NR==2{print $3}’ xtbopt.xyz‘ ; sed -i "2c ${
   energy}" xtbopt.xyz
mv xtbopt.xyz $out$
rm $xxx$.xyz $xxx$.out charges wbo xtbopt.log xtbrestart
>>>>
```

You can see that we put gold atoms randomly in a sphere of radii 5,5,5, and put ligands in a micelle structure on a sphere of radii 6,6,6. This sphere is larger because Au-P bond is relatively long. We will explain the commands between two `>>>>` in Subsection 2.4.3.

With these files, run the global optimization:

```
nohup lego au8pph37.inp > au8pph37.out &
```

The global minimum is `au8pph37-LM/58.xyz`, shown in Figure 2.4.1.

2.4.2 Example 13: $Co_6Te_8(PEt_3)_6$

You can find all input and output files in `testfiles/lego/13-co6te8pet36`.

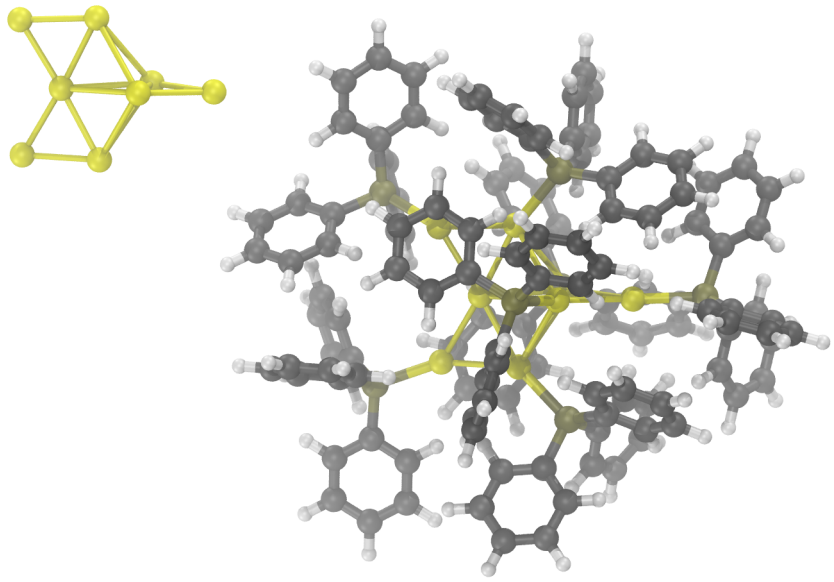


Figure 2.4.1: The global minimum of $Au_8(PPh_3)_7^{2+}$, with the gold core highlighted.

In this section, we want to search another ligand protected cluster $Co_6Te_8(PEt_3)_6$.

Since we do not know the structure of Co_6Te_8, we can do this search first. You can use `isomer` with xTB with the following file `co6te8.inp`:

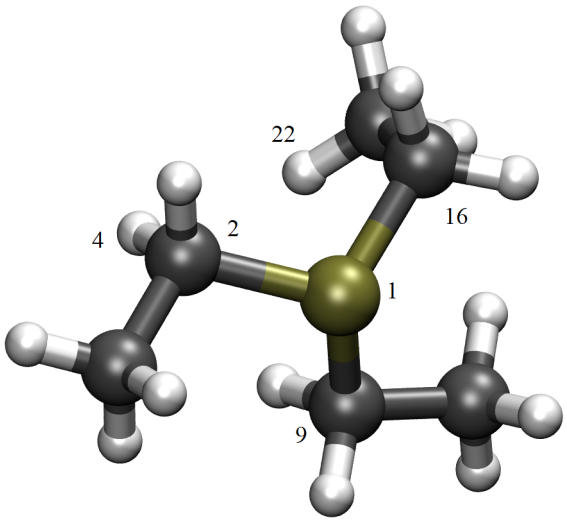
```
co6te8          # Result file name
Co 6 Te 9       # Symbols
cube 4 4 4      # Structure types
100             # Maximal number of calculations
>>>>
cp $inp$ $xxx$.xyz
../xtb/bin/xtb $xxx$.xyz -gfn 1 -o > $xxx$.out
energy=‘awk ’NR==2{print $3}’ xtbopt.xyz‘ ; sed -i "2c ${
   energy}" xtbopt.xyz
mv xtbopt.xyz $out$
rm $xxx$.xyz $xxx$.out charges wbo xtbopt.log xtbrestart
>>>>
```

Run the global optimization with `isomer co6te8.inp > co6te8.out`. The global minimum is `co6te8-LM/40.xyz` and is shown in Figure 2.4.2.

Use this structure and `pet3.xyz`

```
 22
PEt3
 P                    0.77411167    0.93908628    0.00000000
 C                    1.38078883    1.79703460   -1.48602405
 H                    1.02413441    2.80584461   -1.48602405
 H                    2.45078883    1.79702166   -1.48602419
 C                    0.86744661    1.07107703   -2.74342827
 H                    1.19588317    1.59486033   -3.61674408
 H                    1.25178947    0.07263744   -2.76071851
 H                   -0.20200553    1.04133445   -2.72647284
 C                    1.58017719   -0.68723832    0.13314147
 H                    1.06926441   -1.39066966   -0.49059924
```

```
 H                  2.59983880   -0.60671904   -0.18103498
 C                  1.53001448   -1.16683089    1.59569905
 H                  1.95386617   -2.14689639    1.66441129
 H                  2.08820972   -0.49286368    2.21139887
 H                  0.51311419   -1.19395588    1.92747918
 C                  1.16337141    1.93168334    1.47500099
 H                  0.62390995    1.54801668    2.31564434
 H                  0.88119749    2.94963161    1.30453342
 C                  2.67528550    1.85868155    1.75852616
 H                  3.05363867    2.84440501    1.93201431
 H                  2.84706388    1.25327881    2.62390451
 H                  3.17563928    1.42763883    0.91665428
```



and `co6te8pet36.cluster`

```
2
co6te8.xyz    1
pet3.xyz      6
```

We can do the search. The input file `co6te8pet36.inp` is:

```
co6te8pet36             # Result file name
co6te8pet36.cluster     # Cluster file name
100                     # Maximal number of calculations
>>>>
pos 0 0 0 0 0 0
micelle 1 1 4 22 0 0 0 4 4 4
>>>>
cp $inp$ $xxx$.xyz
../../xtb/bin/xtb $xxx$.xyz -gfn 1 -o > $xxx$.out
energy=‘awk ’NR==2{print $3}’ xtbopt.xyz‘ ; sed -i "2c ${
   energy}" xtbopt.xyz
mv xtbopt.xyz $out$
rm $xxx$.xyz $xxx$.out charges wbo xtbopt.log xtbrestart
>>>>
```

You can see that we fix the core and let ligands distribute around it. We let the bonding direction parallel to atom 1 and the center of atom 4 and 22. You can also use `micelle 1 1 1 2 9 16` (the center of three carbon atoms).

Run the global optimization:

```
nohup lego co6te8pet36.inp > co6te8pet36.out &
```

The global minimum is `co6te8pet36-LM/65.xyz`, shown in Figure 2.4.2. All ligands are bonding with cobalt atoms.

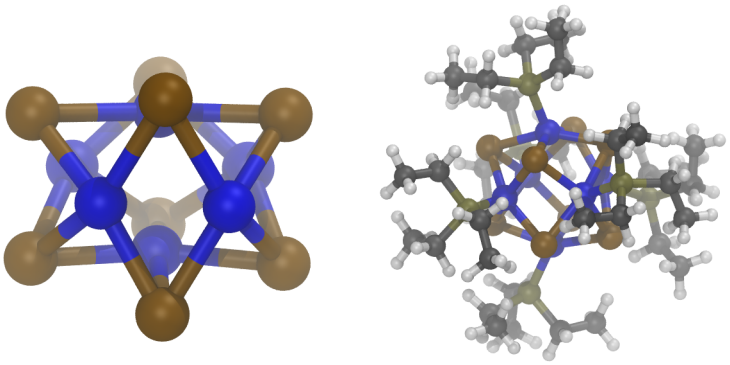


Figure 2.4.2: The global minimum of Co_6Te_8 and $Co_6Te_8(PET_3)_6$.

2.4.3 Theoretical Background

First we explain how NWPEsSe works with xTB. Look at the commands below:

```
cp $inp$ $xxx$.xyz
../xtb/bin/xtb $xxx$.xyz -gfn 1 -chrg +2 -o > $xxx$.out
energy=‘awk ’NR==2{print $3}’ xtbopt.xyz‘ ; sed -i "2c ${energy}" xtbopt.xyz
mv xtbopt.xyz $out$
rm $xxx$.xyz $xxx$.out charges wbo xtbopt.log xtbrestart
```

The first line copies the geometry to a XYZ file for xTB. The second line optimizes the geometry (`-o`). xTB will store the optimized structure in a file called `xtbopt.xyz`. Thus, the third line changes the title to its energy. The fourth line copies `xtbopt.xyz` to `$out$`. The fifth line deletes the redundant files.

Below are some tips for efficient use of NWPEsSe with xTB.

- The charge and spin of the system can be adjusted with the keyword `-chrg` and `-u`
- The current version of xTB uses GFN2 Hamiltonian (*J. Chem. Theory Comput.* **2019**, *15*, 1652) as default . However, in some cases, especially atomic clusters, the calculation may have difficulty in convergence. Thus, you may use GFN1 Hamiltonian (*J. Chem. Theory Comput.* **2017**, *13*, 1989) by using `-gfn 1` as shown above. For organic molecular clusters, GFN2 works perfectly and can be safely used.
- For complex systems, run more global optimizations to get reliable results.
- In some cases, it is needed to adjust energy boundary for more efficient global optimization. See Section 2.10 for details.

2.5 lego with CP2K

To study clusters in periodic systems, like Au_8 on TiO_2 surface, first principle programs are needed. In this section we are going to use CP2K, a free and powerful program.

2.5.1 Example 14: Au_8 on O-Defected Rutile (110) Surface

You can find all input and output files in `testfiles/lego/14-au8tio2110`.

Rutile and gold clusters play important roles in catalysis. In this section, we want to search the global minimum of rutile surface-supported Au_8. The Miller index of the selected surface is 110, with an O-defect. It will provide extra electrons to the gold cluster.

First prepare the gold atom and TiO2(rutile, 110, O-defect) surface. `au.xyz`:

```
1
Au
Au  0.  0.  0.
```

`tio2110.xyz`. You can find it in `testfiles/lego/14-au8tio2110/tio2110.xyz`. It contains 4 layers of TiO_2. The cell lengths for X and Y directions are 18.91 and 17.72 Å, respectively. You can find an O-defect on the surface.

Prepare the cluster file `au8tio2110.cluster`:

```
2
tio2110.xyz     1
au.xyz          8
```

Prepare the input file `au8tio2110.inp`:

```
au8tio2110              # Result file name
au8tio2110.cluster      # Cluster file name
50                      # Maximal number of calculations
>>>>
pos 9.455 8.86 0. 0. 0. 0.
inbox 7. 5. 9. 14. 12. 10.
>>>>
cp2k4nwpesse.sh $xxx$ $inp$ $out$
>>>>
```

Note that, the surface was fixed at its center with no rotations (If rotated, the periodic boundary condition will be destroyed!) The box defined in the second line lies above the O-defect.

The command `cp2k4nwpesse.sh $xxx$ $inp$ $out$` calls CP2K for NWPEsSe. We will describe them now.

You need to copy two files to the current directory from `misc/cp2k4nwpesse`: `geoopt.tmp` and `cp2k4nwpesse.sh`. The former is a template for geometry optimization using CP2K, and the latter is the script to call CP2K.

You need to change both for NWPEsSe. First, for `geoopt.tmp`, you can find the following lines:

```
PROJECT_NAME
...
WFN_RESTART_FILE_NAME
...
@INCLUDE  XXX
```

```
...
```

NEVER change the three lines!

You can change other places for your calculation. You should give the correct basis set, pseudopotential, and DFT-D parameters file names:

```
BASIS_SET_FILE_NAME    ../BASIS
POTENTIAL_FILE_NAME    ../GTH_POTENTIALS
...
PARAMETER_FILE_NAME  ../dftd3.dat
```

Note that `BASIS` and `GTH_POTENTIALS` can be found in the CP2K distribution. Thus, the following lines are also needed to be changed according to the elements appeared in your cluster:

```
&KIND H
  BASIS_SET  DZVP-MOLOPT-SR-GTH-q1
  POTENTIAL  GTH-PBE-q1
&END KIND
&KIND Ti
  BASIS_SET  DZVP-MOLOPT-SR-GTH-q12
  POTENTIAL  GTH-PBE-q12
&END KIND
```

You can change the functional name:

```
&XC_FUNCTIONAL NO_SHORTCUT
 &PBE T
 &END PBE
&END XC_FUNCTIONAL
```

And most importantly, the cell parameters:

```
ABC      18.91 17.72 30.00
PERIODIC XYZ
```

Note that since we are studying surfaces, a vacuum space is needed between the TiO_2 slabs, so the Z cell length is 30 Å. Therefore, a dipole correction has to be applied:

```
SURFACE_DIPOLE_CORRECTION T
SURF_DIP_DIR              Z
```

In this study, we freeze the last two layers of TiO_2 by the following commands:

```
&CONSTRAINT
  &FIXED_ATOMS
    COMPONENTS_TO_FIX XYZ
    LIST  216..431
  &END FIXED_ATOMS
&END CONSTRAINT
```

You can delete these in your own works if you do not need them.

Second, change `cp2k4nwpesse.sh`. You need to change **only** one line:

```
mpirun -n 144 cp2k.popt -i ${calcinpfn} -o ${calcoutfn}
```

Usually, CP2K calculations are done with multi-node parallelization. In my case, I used 144 CPU cores. You need to change this for your cluster. For example, `mpirun` may have to be changed to `srun` for a slurm queueing system.

With these files, run the global optimization:

```
nohup lego au8tio2110.inp > au8tio2110.out &
```

The global minimum is `au8tio2110-LM/49.xyz`, shown in Figure 2.5.1.

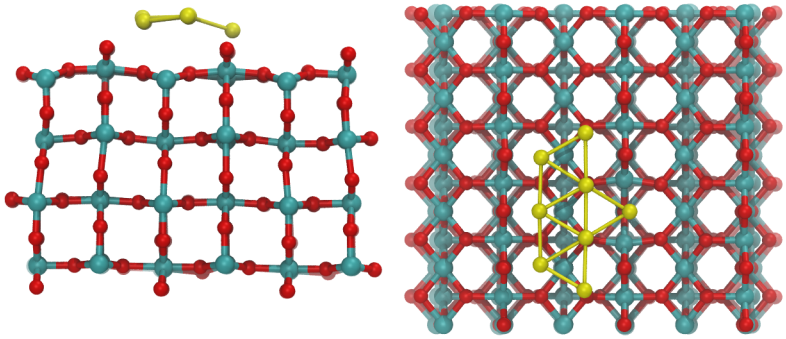


Figure 2.5.1: The global minimum of Au_8 on O-defected rutile (110) surface.

Actually, from `au8tio2110.out` we know that there are two clusters having unusual lower energy: `au8tio2110-LM/42.xyz` and `au8tio2110-LM/46.xyz`. But in these two clusters, the surfaces are highly distorted, so I do not think it is related to our problems. However, if you think they are indeed what you need, accept them.

2.5.2 Theoretical Background

- CP2K calculations are very expensive. Think carefully before you start production runs.
- For surface-supported clusters, `pos` is always preferred to be applied to the surface. The remaining atoms or molecules are distributed in a box above the surface. Always set correct `ABC` parameters in `geoopt.tmp`. The surface slab should be sufficiently large. Dipole correction has to be applied (`SURFACE_DIPOLE_CORRECTION`).
- There may be some highly distorted clusters having very low energies. You should decide by yourself whether they are reasonable in your applications.
- If transition or lanthanide atoms are present, the cutoff of plane waves should be larger than 500 (using `CUTOFF 500`).
- For different surfaces, like graphene, rutile, or platinum, different DFT calculation schemes are required. For example, for graphene or rutile, `OT` is an effective algorithm; for platinum, exact diagonalization may be better. Do calibrations before large-scale calculations.

2.6 `lego` with Gromacs

Gromacs is a very efficient molecular dynamics program. We will use it to optimize very large clusters of soft matter. Note that in such cases, the true global minimum is almost impossible to find. However, using NWPEsSe, we can get a very stable initial configuration for further study of such systems.

2.6.1 An Additional Input Option in inp File

In this section, we introduce an option in the NWPEsSe input file. It is rarely used. But when very large molecules are involved in the global optimization. For every guess cluster, NWPEsSe will do a pre-optimization before submitting it to third-party programs. However, for very large systems (hundreds of very large molecules, *vide infra*), the pre-optimization may be as expensive as Gromacs calculations. Thus, we can skip this pre-optimization when `lego` works with Gromacs. To do this, add `N` at the last line of `*.inp` files.

For example, if we generate 4 water molecules in a box, the difference between adding `N` and not adding `N` is shown in Figure 2.6.1.

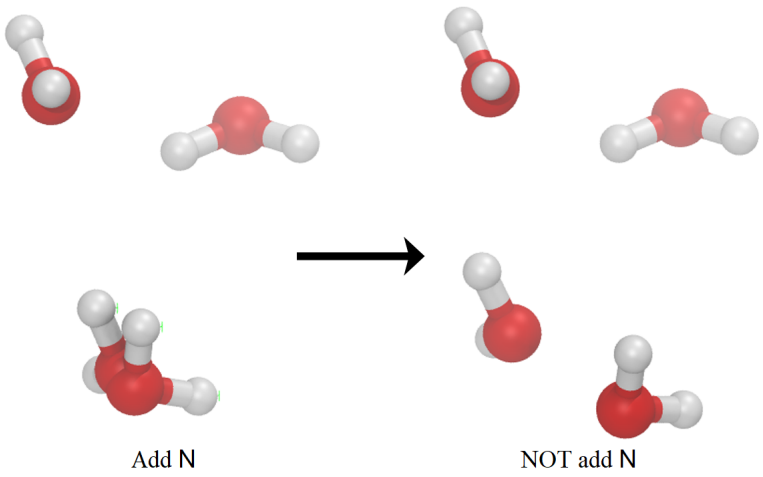


Figure 2.6.1: The difference between adding `N` and not adding `N` in generating 4 water molecules in a box.

Again: in most other cases do **NOT add N** unless you are sure what you are doing.

2.6.2 Example 15: A Reversed Micelle of POPC

You can find all input and output files in `testfiles/lego/15-reversedmicelle`.

In this section, we want to get some reversed micelle-like cluster of water and POPC. The cluster will be optimized using Gromacs. **Readers are assumed to be familiar with Gromacs.**

For cluster optimization with Gromacs, force field parameters are required. They can be found in `testfiles/lego/15-reversedmicelle/gromacs_tops`. In these files, the residue names of water and POPC are `SOL` and `POPC`, respectively. We consider a system of 300 water and 100 POPC molecules. Thus, the essential part of the topology file `testfiles/lego/15-reversedmicelle/system.top` is:

```
[molecules]
POPC 100
SOL  300
```

The molecular dynamics parameters file is `testfiles/lego/15-reversedmicelle/min.mdp`. We use steepest decent algorithm to minimize the system for 20000 steps.

```
integrator              = steep
nsteps                  = 20000
```

The geometries of water and POPC are `testfiles/lego/15-reversedmicelle/spce.xyz` and `testfiles/lego/15-reversedmicelle/popc.xyz`, respectively. Pay attention that unlike ordinary XYZ files used before, **the element names in XYZ files for NWPEsSe+Gromacs must be the same as those in topology files.** For example, in `testfiles/lego/15-reversedmicelle/spce.xyz`

```
3
water
      OW    12.47    38.07    55.20
     HW1    11.47    38.06    55.24
     HW2    12.77    38.49    54.35
```

`OW` has to be used instead of `O`.

The cluster file `testfiles/lego/15-reversedmicelle/reversedmicelle.cluster` is

```
2
popc.xyz    100
spce.xyz    300
```

The input file `testfiles/lego/15-reversedmicelle/reversedmicelle.inp` is

```
reversedmicelle           # Result file name
reversedmicelle.cluster   # Cluster file name
20                        # Maximal number of calculations
>>>>
micelle 4 13 50 50 50 15 8 8
insphere 50 50 50 14.5 7.5 7.5
>>>>
gromacs4abc-1.py $inp$ system.gro 100 100 100
gmx_d grompp -quiet -c system.gro -f min.mdp -p system.top -
   o system-min.tpr  1>nul 2>nul
gmx_d mdrun -quiet -deffnm system-min  1>nul 2>nul
gromacs4abc-2.py system-min.gro system-min.log $out$
rm -rf system.gro system-min.gro system-min.log system-min.
   tpr system-min.trr system-min.edr mdout.mdp step*.pdb
>>>>
N
```

The strutural features indicate that water molecules are distributed inside an ellipsoid of radii 14.5,7.5,7.5. The POPC molecules are distributed on the surface of a slightly larger ellipsoid, with the hydrophilic ends pointing toward water. Note that the centers are `50 50 50` because we want to put the reversed micelle in the middle of the simulation box.

For the commands, the lines `gmx_d ...` run the optimization, and the line `rm -rf ...` deletes the redundant files.

Two Python files are needed for the transformation between XYZ and GRO file formats. They can be copied from `misc/gromacs4nwpesse`. You must have Python installed on your computer. For `gromacs4ab-2.py`, nothing needs to be changed.

For `gromacs4ac-1.py`, the last thress arguments `100 100 100` are the Gromacs simulation box size (in Å! **NOT** nm in Gromacs). Also, you need to open it and change the following part:

```
residues = [
```

```
    ("POPC", 100, 52),
    ("SOL",  300, 3)
    ]
```

Each line is a tuple containing the residue name, number of the residues, and the number of atoms in the residue. You need to change them for your system.

Now you can run the global optimization:

```
nohup lego reversedmicelle.inp > reversedmicelle.out
```

You will get 20 reversed micelles in `reversedmicelle-LM`. Figure 2.6.2 shows one example.

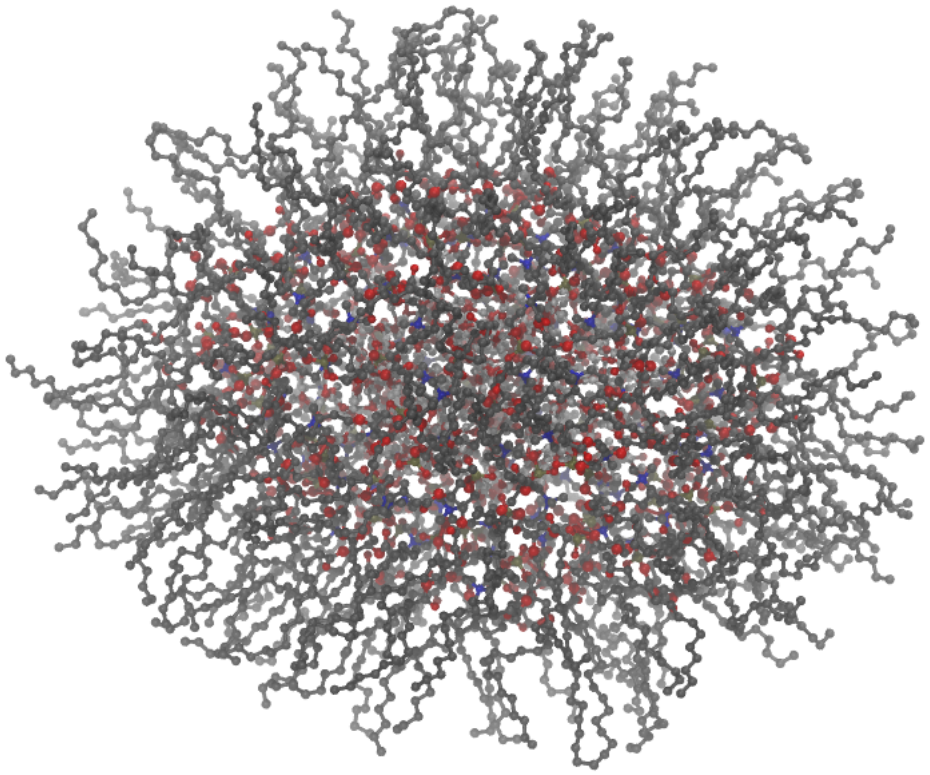


Figure 2.6.2: One reversed micelle generated by `lego`.

2.6.3 Example 16: Two Layers of POPC

You can find all input and output files in `testfiles/lego/16-layers`.

We will create two layers of POPC and optimize the membrane with Gromacs.

The cluster file `testfiles/lego/16-layers/layers.cluster`:

```
2
popc.xyz    225
popc.xyz    225
```

The corresponding part of `gromacs4abc-1.py` should be changed:

```
residues = [
    ("POPC", 225, 52),
    ("POPC", 225, 52)
    ]
```

The input file `testfiles/lego/16-layers/layers.inp`:

```
layers              # Result file name
layers.cluster      # Cluster file name
20                  # Maximal number of calculations
>>>>
layer 4 13 0 0 0  1 2 2 45 2 100 45 100 2 45
layer 4 13 0 0 0 -1 2 2 95 2 100 95 100 2 95
>>>>
gromacs4abc-1.py $inp$ system.gro 100 100 130
gmx_d grompp -quiet -c system.gro -f min.mdp -p system.top -
   o system-min.tpr  1>nul 2>nul
gmx_d mdrun -quiet -deffnm system-min  1>nul 2>nul
gromacs4abc-2.py system-min.gro system-min.log $out$
rm -rf system.gro system-min.gro system-min.log system-min.
   tpr system-min.trr system-min.edr mdout.mdp step*.pdb
>>>>
N
```

We can see that `0 0 1` and `0 0 -1` make the POPC pointing in opposite directions.

Now run the global optimization:

```
nohup lego layers.inp > layers.out
```

You will get 20 clusters in `layers-LM`. Figure 2.6.3 shows one example.

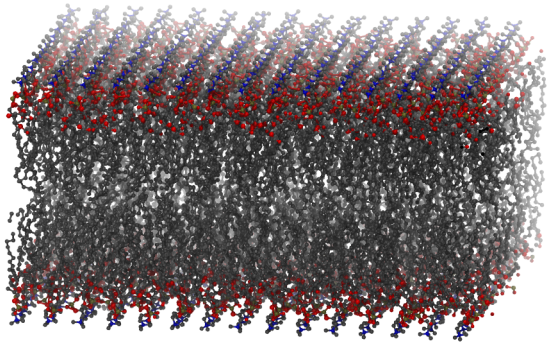


Figure 2.6.3: One cluster generated by `lego`.

Note that this double-layer is only an illustration example. It is **NOT** a real membrane used in biological studies.

If you need single layer, you can remove one component from the clusters.

2.6.4 Theoretical Background

- The cluster structures obtained in these calculations must be equillibrated in solvents for some time (for membranes, at least 10 ns) before production molecular dynamics runs.

- The simulation box size must match the periodicity of the system.

2.7 `lego` with Lammps

Here, Lammps is used with `lego` to do global optimizations for the reactive force field ReaxFF.

2.7.1 Example 17: $Ni_{38}(CO)_{36}$

You can find all input and output files in `testfiles/lego/17-ni38co36`.

Some experiments implied that small nickel cluster, Ni_{38} can activate CO (*J. Chem. Phys.*, **1997**, *107*, 1861). Saturation coverage was observed as ceNi38(CO)36. We want to find the global minimum of this. The interaction is described with ReaxFF. **Readers are assumed to be familiar with Lammps.**

Prepare the stucture of Ni_{38} and CO:

`ni38.xyz`

```
38
Ni38
 Ni       6.30445000      1.93201000     -2.23037000
 Ni       4.85996000      5.16445000     -1.15818000
 Ni       6.70744000      1.23898000      0.22006600
 Ni       5.74416000      9.54662000     -0.22006600
 Ni       5.09216000      7.62167000     -1.89751000
 Ni       3.20439000      5.88555000      0.65338600
 Ni       5.28009000      4.53206000      1.28405000
 Ni       6.14715000      8.85359000      2.23037000
 Ni       6.51910000      4.43722000     -2.96845000
 Ni       9.24721000      4.90005000     -0.65338600
 Ni       6.93911000      3.78532000     -0.53140300
 Ni       8.63026000      3.07251000     -2.37730000
 Ni       9.50536000      7.37548000     -1.42336000
 Ni       8.03966000      4.98459000      3.63355000
 Ni       7.59164000      5.62115000      1.15818000
 Ni       8.85878000      5.54893000     -3.13491000
 Ni       7.40243000      8.75335000     -2.04729000
 Ni       4.19082000      3.32801000     -2.88128000
 Ni       9.71249000      4.25973000      1.80483000
 Ni       4.64124000      2.70675000     -0.40732300
 Ni       9.93840000      6.73257000      1.05351000
 Ni       7.17151000      6.25354000     -1.28405000
 Ni       4.41194000      5.80101000     -3.63355000
 Ni       2.51320000      4.05303000     -1.05351000
 Ni       8.26078000      7.45759000      2.88128000
 Ni       5.93250000      6.34838000      2.96845000
 Ni       9.03050000      2.40846000      0.08907010
 Ni       5.04917000      2.03225000      2.04729000
 Ni       3.42110000      8.37714000     -0.08907010
 Ni       7.81036000      8.07885000      0.40732300
 Ni       3.82134000      7.71309000      2.37730000
 Ni       3.59282000      5.23667000      3.13491000
 Ni       6.75212000      6.91106000     -3.75579000
 Ni       5.51249000      7.00028000      0.53140300
 Ni       2.94624000      3.41012000      1.42336000
```

```
 Ni       5.69948000       3.87454000       3.75579000
 Ni       2.73911000       6.52587000      -1.80483000
 Ni       7.35944000       3.16393000       1.89751000
```

co.xyz

```
2
CO
  C      6.22580000       5.39280000      -0.64315500
  O      6.22580000       5.39280000       0.64315500
```

The cluser file `ni38co36.clsuter`

```
2
ni38.xyz    1
co.xyz      36
```

The input file `ni38co36.inp`

```
ni38co36             # Result file name
ni38co36.cluster     # Cluster file name
1000                 # Maximal number of calculations
>>>>
pos 0. 0. 0. 0. 0. 0.
onsurface 1 1 2
>>>>
lammps4abc-1.py $inp$ data.abc -20. +20. -20. 20. -20. +20.
   0 0 0 3 Ni C O
set NUMBER_OF_PROCESSORS=8 & lmp_serial < in.abc > out.abc
   2>nul
lammps4abc-2.py dump.abc.xyz out.abc $out$ 3 Ni C O
rm -rf log.cite log.lammps data.abc out.abc dump.abc.xyz
>>>>
```

In the structural feature part, the nickle cluster is fixed in space, and CO molecules are distributed on its surface.

In the command part, `lmp_serial ...` runs the Lammps calculation, `rm ...` deletes the redundant files. Three other files are needed: `in.abc`, `lamps4abc-1.py`, and `lamps4abc-2.py`. They can be copied from `misc/lammps4nwpesse`. Also, a ReaxFF parameter file is needed. It is called `CHOSMoNiLiBFPN.ff` (*J. Electrochem. Soc.* **2014**, *161*, E3009), which can be found in `testfiles/lego/17-ni38co36`.

In `in.abc`, change the following line according to your case. This line suggests the force field you use.

```
pair_coeff  * * CHOSMoNiLiBFPN.ff Ni C O      # !!!! Change this line for your ReaxFF fo
```

For `lammps4abc-*.py`, nothing needs to be changed in them. You have to change some arguments. For `lammps4abc-1.py`, the 3rd to 9th arguments are `xl,xh,yl,yh,zl,zh,xy,xz,yz` parameters in Lammps. Then, you need to write the number of elements and their element symbols. For `lammps4abc-2.py`, the 4th argument is also the number of eleents, followed by their element symbols.

Now run the global optimization:

```
nohup lego ni38co36.inp > ni38co36.out
```

You will get 1000 clusters in `ni38co36-LM`.

Note that `onsurface` will put CO randomly on the surface. However, we can also try `micelle`, which put CO uniformly on the surface:

```
micelle 1 2 0 0 0 5 5 5
```

Figure 2.7.1 shows some examples. It seems that clusters obtained by `micelle` are more stable. Of course, you should determine by yourself that which one (randomly vs. uniformly) is needed in your research.

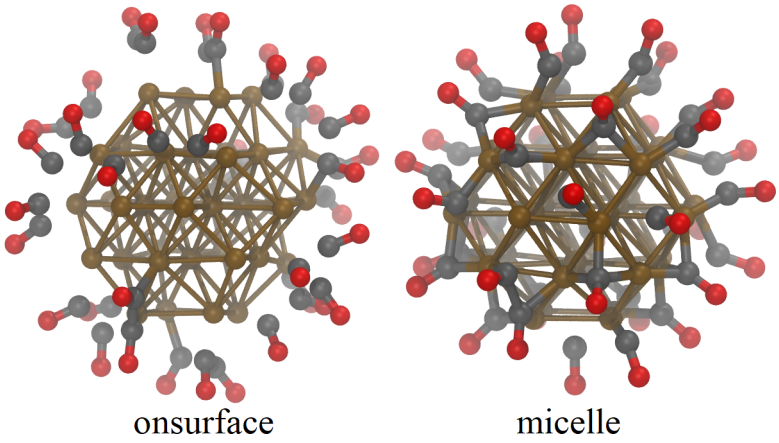


Figure 2.7.1: Typical clusters of $Ni_{38}(CO)_{36}$ obtained by `onsurface` and `micelle`.

2.7.2 Theoretical Background

- Choose the suitable ReaxFF force field. Even you have a parameter file "CHSN.ff", it is possible that some cross-element cases, say C-S, are not well trained. Always check the original paper for this.
- ReaxFF has a multi-modal functional form. A large number of steps is needed.

2.8 `lego` with NWChem, ORCA, MOPAC

The script `misc/nwchem4nwpesse.sh`, `misc/orca4nwpesse.sh`, and `misc/mopac4nwpesse.sh` can be used to combine NWPEsSe with NWChem, ORCA, and MOPAC, respectively. They can be used in `lego` in the same way. For example:

```
aul                # Result file name
aul.cluster        # Cluster file name
1000               # Maximal number of calculations
>>>>
insphere 0 0 0 5 5 5
micelle 1 1 1 2 12 23 0 0 0 6 6 6
>>>>
mopac4nwpesse.sh $xxx$ $inp$ $out$
>>>>
```

Of course, ***.sh must be in PATH variable, or be called with path.**

2.9 lego with DMol3

The script misc/dmol34nwpesse.sh can be used to combine NWPEsSe with DMol3 (Courtesy of Prof. Dr. Lei Ma and Mr. Kai Wang, Tianjin International Center for Nanoparticles and Nanosystems (TICNN), http://ticnn.tju.edu.cn).)

2.10 Energy Boundary Setting

When lego generates a cluster, the optimized structure will be read by lego and is used to generate new guesses. Of course, the ones with lower energy are better candidates. However, some bad structures can have low energies due to the deficiency of computational chemical methods. In this case, it is favourable to set an energy boundary. A cluster that has an energy outside the boundary will be discarded, making global optimization more efficient.

2.10.1 Example 18: $EMIM_3Cl_5^{2-}$

You can find all input and output files in testfiles/lego/18-emimcl.

In this subsection, we want to find the global minimum of an ionic liquid cluster $EMIM_3Cl_5^{2-}$, which is composed of 3 $EMIM^+$ and 5 Cl^-. Since the cation and anion are very regular organic/inorganic molecules, it is very reasonable to combine lego with xTB.

Prepare the structures of $EMIM^+$ and Cl^-.

emim.xyz:

```
19
EMIM(-)
N 2.187469 -1.660192 -0.353068
C 1.757536 -1.644943 0.914236
N 2.232132 -0.542047 1.501845
C 2.953898 0.173011 0.586948
C 2.925068 -0.532309 -0.586489
C 1.840081 -2.674052 -1.355483
C 1.066029 -2.065377 -2.509750
C 1.870399 -0.072180 2.834270
H 1.086623 -2.352958 1.366123
H 3.400887 1.120444 0.826002
H 3.339576 -0.302858 -1.550584
H 2.769423 -3.133339 -1.691155
H 1.251164 -3.428139 -0.838516
H 0.822941 -2.851009 -3.223815
H 1.647546 -1.300251 -3.019763
H 0.146856 -1.608473 -2.156579
H 1.175041 0.756299 2.730042
H 2.772643 0.239439 3.353475
H 1.379866 -0.878443 3.367685
```

cl.xyz:

```
1
Cl(-)
Cl          0.00000          0.00000          0.00000
```

Prepare the cluster file `emimcl.cluster`:

```
2
emim.xyz     3
cl.xyz       5
```

Now we are going to prepare `emimcl.inp`.

```
emimcl            # Result file name
emimcl.cluster    # Cluster file name
500               # Maximal number of calculations
>>>>
inbox 0 0 0 10 10 10
inbox 0 0 0 10 10 10
>>>>
cp $inp$ $xxx$.xyz
../xtb $xxx$.xyz -chrg -2 -o > $xxx$.out
energy=`awk 'NR==2{print $2}' xtbopt.xyz` ; sed -i "2c ${
   energy}" xtbopt.xyz
mv xtbopt.xyz $out$
rm $xxx$.xyz $xxx$.out charges wbo xtbopt.log xtbrestart
>>>>
```

Note that we write `-chrg -2` to tell xTB that the system has two negative charges.

With these files, run the global optimization:

```
nohup lego emimcl.inp > emimcl.out &
```

In the output file `emimcl.out`, we can find this:

```
Reordered from low to high energy:
================================================================

    #           Energy
================================================================

  466     -96.29691227
  472     -96.29679381
...
  185     -96.27701445
  314     -96.27661512
  385     -96.27625570
   34     -96.27581349
  481     -96.27557124
  384     -96.27543809
```

However, these low energy clusters are not bound ones. Due to the large Coulomb repulsion, one Cl^- is pushed away, say `385.xyz`. A bound cluster thus may have a higher energy. For example, a bound cluster `384.xyz` is $(-96.27543809) - (-96.27625570) = 0.00081$ Hartree higher than `385.xyz`. See Figure 2.10.1.

Therefore, `lego` mistakenly treated these low energy yet unbound clusters as good clusters and spent a lot of time in looking for these unbound clusters. In this case, one can add some commands at the last line of `emimcl.inp` to increase the efficiency of global optimization:

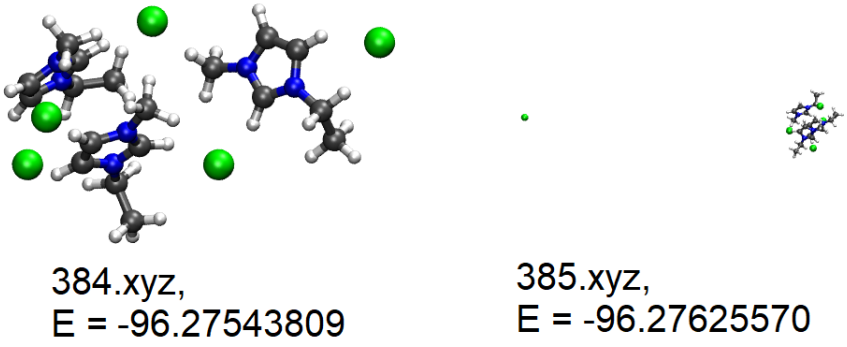


Figure 2.10.1: Two structures of $EMIM_3Cl_5^{2-}$.

```
emimcl            # Result file name
emimcl.cluster    # Cluster file name
500               # Maximal number of calculations
>>>>
inbox 0 0 0 10 10 10
inbox 0 0 0 10 10 10
>>>>
cp $inp$ $xxx$.xyz
../xtb $xxx$.xyz -chrg -2 -o > $xxx$.out
energy=‘awk ’NR==2{print $2}’ xtbopt.xyz‘ ; sed -i "2c ${
   energy}" xtbopt.xyz
mv xtbopt.xyz $out$
rm $xxx$.xyz $xxx$.out charges wbo xtbopt.log xtbrestart
>>>>
Y -96.2760 +100
```

The first character of the last line, `Y` means to do a pre-optimization (see Section 2.6.1). The following two real numbers means that only clusters that have energy in $[-96.2760, +100.0000]$ are treated as good ones. The energy `-96.2760` is obtained from the observation that all clusters having energy lower than -96.2760 are unbound ones, so the desired (bound) clusters should have a higher energy.

Run the global optimization:

```
nohup lego emimcl.inp > emimcl.out &
```

In the output file `emimcl.out`, we can find this:

```
================================================================

    #           Energy
================================================================

   41     -96.27595920
  428     -96.27578187
  185     -96.27566801
  356     -96.27546900
   86     -96.27533139
  461     -96.27522497
```

```
...
```

We can see that, although `41.xyz` and `428.xyz` are still unbound clusters, `185.xyz` is already a bound one (see Figure 2.10.2). This significantly increase the global optimization efficiency and accuracy. This is a highly important strategy for global optimization of complicated clusters.

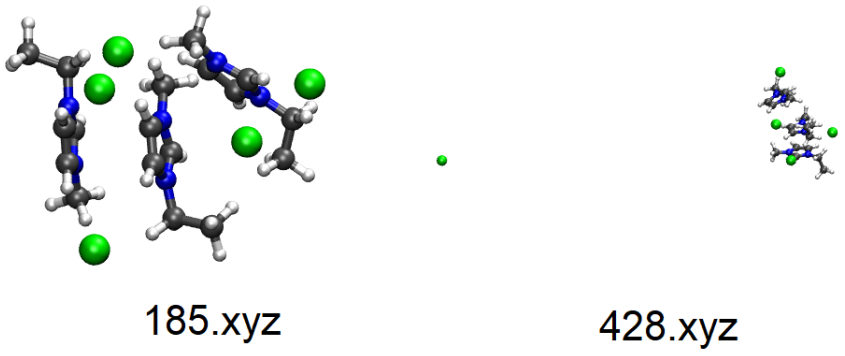


Figure 2.10.2: Two structures of $EMIM_3Cl_5^{2-}$.

2.11 `lego` with More Programs

While I have provided a lot of interfaces, there are of course a lot of other programs for which interfaces are not provided, like ADF, BigDFT, etc. **If you have written one and think it works well, I will be very appreciated if you could send it to me (mailto: ZhangJunQcc@gmail)** thus I can put them in the next release of NWPEsSe. Your contribution will be acknowledged. People share and gain!

Before finishing this chapter, I would like to remind you that it is possible to use more programs at the same time to increase the search efficiency of NWPEsSe. For example, use NWPEsSe, xTB and Gaussian:

```
h2o4io3i2o5              # Result file name
h2o4io3i2o5.cluster      # Cluster file name
50                       # Maximal number of calculations
>>>>
inbox 0. 0. 0. 7. 7. 7.
inbox 0. 0. 0. 7. 7. 7.
>>>>
cp $inp$ $xxx$.xyz
../xtb/bin/xtb $xxx$.xyz -gfn 1 -chrg +2 -o > $xxx$.out
energy=‘awk ’NR==2{print $3}’ xtbopt.xyz‘ ; sed -i "2c ${
   energy}" xtbopt.xyz
mv xtbopt.xyz $inp$
rm $xxx$.xyz $xxx$.out charges wbo xtbopt.log xtbrestart
./xyz2gaussian optfile $inp$ > $xxx$.gjf
g09 < $xxx$.gjf > $xxx$.log 2>/dev/null
./gaussian2xyz $xxx$.log > $out$
>>>>
```

In this case, the cluster guessed by NWPEsSe will first be optimized with xTB and are then submitted to Gaussian for further optimization. Of course, you can also first use NWPEsSe+xTB to search some clusters, then use Gaussian to further optimize the energetically low-lying ones. You can use the strategy you prefer!

Bibliography