ABCluster Manual

Version 2.0

Jun Zhang

Institute for Theoretical Chemistry, University of Cologne

Department of Chemistry, University of Illinois at Urbana–Champaign


ABCluster: A software for the global optimization of atomic and molecular clusters by the artificial bee colony algorithm
Contents

1 Introduction 1
   1.1 What is ABCluster? .............................................. 1
   1.2 Contacting the Author ........................................ 2
   1.3 Program Citation ............................................. 2
   1.4 Copyright and Disclaimer .................................. 2
   1.5 Components of ABCluster ................................. 2
   1.6 Installation of ABCluster ................................. 3
      1.6.1 For Windows Users .................................. 3
      1.6.2 For Linux and Mac OS X Users ...................... 3
      1.6.3 For Mac OS X Users: If you see dyld: Library not loaded . 3
      1.6.4 For OpenMP ........................................... 4
      1.6.5 If you see Incorrect format near ..................... 4
   2 Theory 5
      2.1 Choose Your Cluster and Potential Energy Function ....... 5
      2.1.1 When Must Quantum Chemistry be Used? ............... 5
      2.1.2 When does Force Field Perform Well? .................. 5
      2.1.3 For Atomic Clusters with Force Fields ............... 6
      2.1.4 For Coarse-Grained Particle Clusters with Force Fields 6
      2.2 The Artificial Bee Colony Algorithm .................... 6
      2.3 Analysis of the Results .................................. 7
   3 Atomic Clusters with Force Fields 9
      3.1 Introduction ............................................... 9
      3.2 Example 1: 38 Lennard-Jones Particles .................. 12
      3.3 About abcinp ............................................. 12
      3.4 Example 2: (MgO+O2-)_20 .................................. 15
      3.5 Example 3: Silve and Copper Clusters .................... 16
         3.5.1 Ag_{38} and Cu_{38} ................................. 16
         3.5.2 Ag_{42}Cu_{6} .................................... 17
      3.6 Example 4: Perovskite Clusters (CaTiO_3)_5 ............. 17
      3.7 Example 5: C_{16} and Ge_{16} Clusters .................. 18
   4 Rigid Molecular Clusters with Force Fields 21
      4.1 Introduction ............................................... 21
      4.2 Example 1: (H_2O)_6 ..................................... 22
      4.3 Example 2: Li^+, Na^+ and Cs^+ in (C_6H_6)_6 ............... 24
      4.4 Example 3: (CH_2=CH_2)_{13} and Build New Force Fields 24
      4.5 Example 4: La^{3+}(H_2O)_{15} at Gold {100} Surface ....... 27
## CONTENTS

4.6 Example 5: \((\text{H}_2\text{O})_2(\text{CH}_3\text{OH})_2\) in 1D Environments ........................................ 28  
4.7 Example 6: \(\text{HNO}_3(\text{H}_2\text{O})_{10}\) in an Electric Field ................................. 29  

5 Optimization with Third-party Programs .......................... 33  
5.1 Introduction ................................................................. 33  
5.2 ABCluster with Gaussian .................................................. 33  
  5.2.1 Configuration of Gaussian ......................................... 33  
  5.2.2 Example 1: \(\text{Al}_3\text{O}_4^+\) ........................................ 34  
  5.2.3 More About \texttt{optfile} ........................................... 36  
  5.2.4 Example 2: \(\text{Au}_6\) .............................................. 36  
5.3 ABCluster with Other Programs ......................................... 40  
  5.3.1 How does \texttt{isomer} and \texttt{lego} work? ......................... 40  
  5.3.2 Example 3: ABCluster with ORCA on \text{He}_2\text{Ne}_2 .............. 41  
  5.3.3 Support For ORCA, NWChem, xTB-GFN, and DMol3 .................. 42  
5.4 Examples for \texttt{lego} ..................................................... 42  
  5.4.1 Example 4: \((\text{NH}_3)_4\) ........................................ 43  
  5.4.2 Example 5: \((\text{CO})(\text{H}_2\text{O})_5\) Using xTB-GFN .................. 46  
5.5 Example 6: Absorption .................................................... 47  
5.6 Parallelization ............................................................... 48  
  5.6.1 Example 7: \(\text{Zn}_3\text{S}_3\) with Parallelization .................... 48  
  5.6.2 Example 8: \(\text{Fe}(\text{H}_2\text{O})_2^2+\) with Parallelization .......... 50  

6 Coarse-Grained Particle Clusters ................................. 53  
6.1 Introduction ............................................................... 53  
6.2 Example 1: Why \((\text{HF})_2\) is nonlinear? .......................... 55  
6.3 Example 2: A Charge and Dipole Moments ........................... 58  
6.4 Example 3: PY Clusters .................................................. 59  
6.5 Example 4: A Simple Model of Amphiphilic Molecule .............. 60  
6.6 Visualization of Coarse-Grained Particles ........................ 62  

A Version History ........................................................... 63  

Bibliography ................................................................. 65
Chapter 1

Introduction

1.1 What is ABCluster?

Briefly, ABCluster searches the global as well as the local minima of atomic and molecular clusters.

ABCluster is a software developed by Dr. Jun Zhang to perform the global optimization of atomic and molecular clusters. It is an efficient and user-friendly program. It is designed in a way that non-experts, as well as experts, can perform a global optimization readily without knowing too much about the internal algorithm, being a black-box.

The latest version of ABCluster and related resources can be obtained from the web page:


ABCluster has been applied to a lot of real chemical problems by many authors, please refer to:


ABCluster uses very efficient global optimization algorithm, i.e. “the artificial bee colony algorithm”, which was developed in 2005, to perform the search. The algorithm has been modified and improved specially for chemical problems. It has the following features:

- It can perform global optimization for nonmetallic and metallic atomic clusters, organic molecular clusters, and their mixtures.
- It can not only use internally available force fields, but also work perfectly with any other third-party computational chemistry software.
- It is fully parallelized on a single node (with OpenMP). From ABCluster 2.0, ABCluster is also parallelized on many nodes (with an internal mini-queueing system).
- It is very easy to use! Thus you can focus on your own chemical problems without knowing too much about the internal algorithm!

The author knows that many people are very reluctant to read lengthy and tedious manuals, thus the manual is kept as short as possible.
1.2 Contacting the Author

If you have any bug reports, comments, suggestions or the possibility of cooperation on ABCluster, please feel free to contact to Dr. Jun Zhang via E-mail:

ZhangJunQcc@gmail.com

1.3 Program Citation

For any published works using ABCluster please include the following reference[1, 2]:


1.4 Copyright and Disclaimer

ABCluster is Copyright © 2015-2018 Jun Zhang. ABCluster is free of charge to non-profit academic use. The use of ABCluster in commercial packages is not allowed without a prior written commercial license agreement.

The following codes used by ABCluster have special restrictions:

- **libLBFGS** (see http://www.chokkan.org/software/liblbfgs)
  libLBFGS is the C realization of the limited-memory Broyden–Fletcher–Goldfarb–Shanno (L-BFGS) method[3]. It is Copyright © 2002-2014 Naoaki Okazaki. libLBFGS is distributed under the term of the MIT license.

- **Boost** (see http://www.boost.org)
  Boost is a set of C++ libraries. It is Copyright © 2004-2006 Joe Coder. Boost is distributed under the Boost Software License, Version 1.0.

1.5 Components of ABCluster

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

- **atom-optimizer** Perform the global optimization of atomic clusters using force fields.
- **cg-optimizer** Perform the global optimization of coarse-grained particle clusters using force fields.
- **rigidmol-optimizer** Perform the global optimization of rigid molecular clusters.
- **isomer** Perform the global optimization of clusters with third-party programs for atomic clusters.
- **lego** Perform the global optimization of clusters with third-party programs for both atomic and molecular clusters.
- **abcinp** Generate the input files for **atom-optimizer**.
- **xyz2gaussian** A small program to use ABCluster with Gaussian.
1.6 Installation of ABCluster

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

1.6.1 For Windows Users

Assume you put ABCluster (atom-optimizer.exe, etc) in the directory D:\ABCluster, then open

Start→Control Panel→System or Security→System

then click Advanced System Settings, choose the Environment Variables option. In the System variables list, choose Path, then click Edit. In the editbox, add

;D:\ABCluster

(Don’t forget the semicolon “;” !) at the end of the texts in it, and click OK to confirm it.

1.6.2 For Linux and Mac OS X Users

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

export PATH=$PATH:/home/you/bin/abcluster

and run the command source ~/.bashrc.

1.6.3 For Mac OS X Users: If you see dyld: Library not loaded

In some cases, on Mac OS X when you run ABCluster you may see such an error: dyld: Library not loaded: /usr/local/opt/boost/lib/libboost_system.dylib. This means 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
1.6.4 For OpenMP

ABCluster is parallelized by OpenMP. In default, it will use all the CPU cores of the machine on which it is run. If you want to change this behavior, please set $\texttt{OMP\_NUM\_THREADS}$ variable to the number of CPU cores you want to use.

Now ABCluster is ready, please enjoy it!

1.6.5 If you see Incorrect format near ...

Sometimes, if you prepare an input file on Windows and copy it to a Linux or Mac OS X system, ABCluster may not recognize its format and report an error like Incorrect format near .... In this case, please transform the Windows format into Linux format by $\texttt{dos2unix}$ or using $\texttt{:set fileformat=unix}$ in VIM.
Chapter 2

Theory

2.1 Choose Your Cluster and Potential Energy Function

Before using ABCluster to perform the global optimization of clusters, you should know what “type” of clusters and potential energy functions you need.

Given the geometry $X$ of a cluster, the potential energy function $U(X)$ can describe its potential energy. The energy can, of course, be calculated as accurate as you want by quantum chemistry. However, a global optimization requires a large number of energy calculations. Using quantum chemistry is often too time-consuming. In contrast, force fields are much faster than quantum chemistry and can be used for large-scale global optimization. However, the accuracy of a force field is uncertain, making the interpretation of the results difficult. Therefore one must make a compromise between efficiency and accuracy. Below are some suggestions.

2.1.1 When Must Quantum Chemistry be Used?

In these cases, we strongly recommend the use of quantum chemistry. Thus one should use the isomer and lego components of ABCluster combined with quantum chemistry programs like Gaussian, Molpro, ORCA, MOPAC, etc. Please refer to Chapter 5.

- Inorganic clusters, like Au$_{13}$, Al$_3$O$_4^+$, Co$_2$(CO)$_8$, the interactions between the structural units (Au and Au, Al and O, Co and CO) are almost covalent. No force fields known to the author can calculate their energies accurately. One has to use quantum chemistry.

- Clusters with extra, absent electrons or special spin multiplicities, say (H$_2$O)$_{10}^-$, singlet and triplet CH$_2$(CH$_3$OH)$_5$. Their quantum effects are very strong thus quantum chemistry is unavoidable. Note that “extra, absent electrons” should be understood correctly. For example, sodium cation Na$^+$ is a closed-shell species that can be described quite well by force fields, but sodium atom Na must be treated quantum chemically.

- Clusters containing molecules with other non-classical structures.

2.1.2 When does Force Field Perform Well?

For clusters containing ordinary organic, bioorganic and closed-shell inorganic species, the cluster can be described quite well by force fields like CHARMM, OPLS etc. Thus one should use the rigidmol-optimizer component of ABCluster. Please refer to Chapter 4.
2.1.3 For Atomic Clusters with Force Fields

People have developed force fields for special purposes for some kinds of atomic clusters. ABCluster supports a lot of them. However, you must be very careful in interpreting the optimization results. In Chapter 3 Section 3.1 all the supported force fields are listed. You can use atom-optimizer to do this.

1. **Ionic clusters.** If you are studying clusters containing ions, like \((\text{Na}^+\text{Cl}^-)_{30}\), \((\text{Mg}^{2+}\text{O}^2-)_{20}\), you should use Coulomb-Born-Mayer potentials. See Equation 3.1.1.

2. **Large Metal and alloy clusters.** For the study of something like \(\text{Pt}_{38}, \text{Ag}_{90}\text{Au}_{10}\), Gupta or Sutton–Chen are your choices. The former seems to be better since more parameters are available. See Equation (3.1.11) and (3.1.13). However, for small metallic clusters, like \(\text{Au}_8, \text{Cu}_{10}\), the two potentials are very inaccurate. One must use quantum chemistry methods to search their structures.

3. **Fullerene clusters.** The Girifalco potential is the most suitable one. See Equation (3.1.7). In this potential, the fullerene will be described as a point. If its explicit orientation is required, you will have to use non-bonded potential. See Equation (4.1.1).

4. **Oxides, silicates, and silica-based glasses.** Use Coulomb–Morse–Repulsion potential. See Equation (3.1.5).

5. **C, Si, Ge clusters.** The Tersoff potential (3.1.17) is the best choice since it takes the coordination number into account.

6. **Other model potentials.** ABCluster also supports some model potentials, like Morse potential. See Section 3.1.

Once you confirm the system you need, you can choose the potentials listed above. The parameters of the potentials have to be searched from literature, or fitted by yourself. In the distribution of ABCluster the file atomic-force-field.txt contains some parameters. You can also check ABCluster website for updated ones.

2.1.4 For Coarse-Grained Particle Clusters with Force Fields

From ABCluster 1.5, a new component called cg-optimizer was added to enable the global optimization of coarse-grained particle clusters. These particles are described by rather complex, anisotropic interactions, like dipole-dipole, dipole-quadrupole, or the very expensive Paramonov–Yaliraki potential. By carefully designing the model, it is possible to use ABCluster to study amphiphilic molecules, protein assembling, electrostatic responsing, etc. Please refer to Chapter for examples.

2.2 The Artificial Bee Colony Algorithm

ABCcluster uses the so-called “artificial bee colony” (ABC) algorithm to perform the global optimization. It was proposed by Karaboga in 2005[4] and soon gains a lot of studies and applications in various problems. The ABC algorithm mimics the foraging behavior of bees. ABCluster models this behavior by three kinds of bees: employed bees (EM), onlooker bees (OL) and scout bees (SC). A colony tries to find the best nectar as food source. To do this, EMs perform a first search; OLs based on the knowledge of all EMs perform a further research; SCs have a memory of the previous search results.
and decide which nectars are of low quality and then discard them. After several cycles of search, the colony can find the best nectar.

Now we interpret this model to the algorithm used in ABCluster. For the global optimization of chemical cluster geometry, a trial solution $X$ is the nectar and the smoothed potential energy $U$ is its quality (a lower numeric value indicates a higher quality). The cluster is characterized by its size $N$, the estimated maximum coordinate value $L$ (see Figure 2.2.1), and of course the potential parameters. The parameters needed for the ABC algorithm are the following: the size of the population of trial solutions $SN$, the scout limit $g_{\text{limit}}$, and the maximum cycle number $g_{\text{max}}$. Here are some typical choices:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$SN$</td>
<td>10 to 300 or larger</td>
</tr>
<tr>
<td>$g_{\text{max}}$</td>
<td>100 to 100000 or larger</td>
</tr>
<tr>
<td>$g_{\text{limit}}$</td>
<td>3 to 10</td>
</tr>
<tr>
<td>$L$</td>
<td>1.5 to 3 times larger than system size</td>
</tr>
</tbody>
</table>

![Figure 2.2.1: The graphic illustration of $L$.](image)

For details of the algorithms used in ABCluster, please refer to ABCluster citation paper[1]. In the following chapters, we will give some small systems as examples. In a real scientific research by ABCCluster, you should remember the following tips:

- For large clusters, a better strategy is to perform several optimizations with different initial guesses and compare the final results.
- For clusters with short-ranged or multimodal interactions or with multiple interaction sites (like H$_2$O which has two hydrogen bond acceptor and two donor sites), very large $g_{\text{max}}$ and $SN$ are required to get reliable results, e.g. for some clusters with short-ranged Morse potential, $g_{\text{max}} = 15000$ and $SN = 100$ are required.
- $L$ can be 1.5 to 3 times larger than system size.
- The parameters for atomic clusters can be obtained from literature. For molecular clusters, one can construct its force field parameters from CHARMM, OPLS or AMBER. Remember the form of equation (4.1.1) and that unit of $\epsilon$ and $\sigma$ is kJ mol$^{-1}$ and Å, respectively! With wrong units, you will get absurd results. Of course, one can design his own parameters. Please visit the ABCluster website for more information.

### 2.3 Analysis of the Results

After the global optimization by ABCCluster, you will get a global minimum as well as a lot of local minima. Whether the global minimum corresponds to the one in experiment
remains open at this stage. Also, the chemical information may lie not only in the global minimum but also in some local minima. Therefore, please pick up the useful structures by chemical intuition and study them by advanced quantum chemistry!
Chapter 3

Atomic Clusters with Force Fields

3.1 Introduction

The atoms in an atomic cluster can interact with each by various kinds of potentials. Here “atom” can also mean “particle”, like an ion. ABCluster now supports many kinds of potentials. We have collected a lot of parameters in atomic-force-field.txt. If there is no what you need, you have to search in the literature or fit the parameters by yourself.


\[ U_{\text{CBM}} = \sum_{i=1}^{N} \sum_{i<j}^{N} \left( \frac{e^2 q_i q_j}{4\pi\epsilon_0 r_{ij}} + B_{ij} \exp\left(-\frac{r_{ij}}{\rho_{ij}}\right) \right) \]  

(3.1.1)


\[ U_{\text{LJ}} = \sum_{i=1}^{N} \sum_{i<j}^{N} 4\epsilon_{ij} \left( \left( \frac{\sigma_{ij}}{r_{ij}} \right)^{12} - \left( \frac{\sigma_{ij}}{r_{ij}} \right)^{6} \right) \]  

(3.1.2)

3. Coulomb–Lennard-Jones potential (CLJ_N). Includes both the Coulomb and dispersion interaction.

\[ U_{\text{LJ}} = \sum_{i=1}^{N} \sum_{i<j}^{N} \left( \frac{q_i q_j}{r_{ij}} + 4\epsilon_{ij} \left( \left( \frac{\sigma_{ij}}{r_{ij}} \right)^{12} - \left( \frac{\sigma_{ij}}{r_{ij}} \right)^{6} \right) \right) \]  

(3.1.3)


\[ U_{\text{M}} = \sum_{i=1}^{N} \sum_{i<j}^{N} c_{ij} \left( \exp\left(-n\beta_{ij} (r_{ij} - r_{ij}^0)\right) \right) - n \exp\left(-\beta_{ij} (r_{ij} - r_{ij}^0)\right) \]  

(3.1.4)

5. Coulomb–Morse–Repulsion potential[8] (CMR_N). This is very useful in the simulation of oxides, silicates, and silica-based glasses.

\[ U_{\text{CMR}} = \sum_{i=1}^{N} \sum_{i<j}^{N} \left( \frac{q_i q_j}{r_{ij}} + D_{ij} \left( \exp\left(-2\alpha_{ij} (r_{ij} - \rho_{ij})\right) \right) + 2 \exp\left(-\alpha_{ij} (r_{ij} - \rho_{ij})\right) \right) \]  

(3.1.5)
6. Z potential\[9\] (\(Z_N\)). This is inspired by an earlier potential proposed by Dzugutov[10, 11] which was designed to study glass formation. Unlike CBM, LJ and M potentials, they contain minima as well as maxima, which make the close-packing energetically unfavorable and lead to amorphous structures.

\[
U_Z = \sum_{i=1}^{N} \sum_{i<j}^{N} \left( a e^{ar_{ij}} \cos (2kF r_{ij}) + b \left( \frac{\sigma}{r_{ij}} \right)^m + V_0 \right) \theta (r_c - r_{ij})
\]  

(3.1.6)


\[
U_{Gi} = \sum_{i=1}^{N} \sum_{i<j}^{N} \left( -\alpha \left( \frac{1}{s_{ij} (s_{ij} - 1)^3} + \frac{1}{s_{ij} (s_{ij} + 1)^3} - \frac{2}{s_{ij}^4} \right) + \beta \left( \frac{1}{s_{ij} (s_{ij} - 1)^9} + \frac{1}{s_{ij} (s_{ij} + 1)^9} - \frac{2}{s_{ij}^{10}} \right) \right)
\]

\[
s_{ij} = \frac{r_{ij}}{2d}
\]

\[
\alpha = \frac{N_A^2 A}{12 (2d)^6}
\]

\[
\beta = \frac{N_B^2 B}{90 (2d)^{12}}
\]

(3.1.7)

(3.1.8)

(3.1.9)

(3.1.10)


\[
U_G = \sum_{i=1}^{N} \left( \sum_{j=1}^{N} A_{ij} \exp \left( -p_{ij} \left( \frac{r_{ij}}{d_{ij}} - 1 \right) \right) - \sqrt{\rho (r_i)} \right)
\]

(3.1.11)

where:

\[
\rho (r_i) = \sum_{j=1}^{N} \sum_{j\neq i}^{N} c_{ij}^2 \exp \left( -2q_{ij} \left( \frac{r_{ij}}{d_{ij}} - 1 \right) \right)
\]

(3.1.12)


\[
U_{SC} = \sum_{i=1}^{N} \left( \frac{1}{2} \sum_{j=1}^{N} \epsilon_{ij} \left( \frac{a_{ij}}{r_{ij}} \right)^p - \sqrt{\rho (r_i)} \right)
\]

(3.1.13)

where:

\[
\rho (r_i) = \sum_{j=1}^{N} \sum_{j\neq i}^{N} c_{ij}^2 \left( \frac{a_{ij}}{r_{ij}} \right)^q
\]

(3.1.14)

10. Modified Sutton–Chen potential[15] (SCMJB\(_N\)). In 2015, Januszko and Bose proposed a modified form of Sutton–Chen potential, which shows some improvement in reproducing experimental data:

\[
U_{SCMJBJ} = \sum_{i=1}^{N} \left( \sum_{j=1}^{N} \epsilon_{ij} \left( \frac{a_{ij}}{r_{ij}} \right)^p - \sqrt{\rho (r_i)} \right)
\]

(3.1.15)
where:

\[ \rho (r_i) = \sum_{j=1}^{N} c_{ij}^2 \exp \left( -\alpha_{ij} \left( \frac{r_{ij}}{\theta_{ij}} - 1 \right) \right) \]  \hspace{1cm} (3.1.16)

11. Tersoff potential[16] \((T_N)\). This very complex many-body potential is very useful for describing C, Si, Ge, N, B clusters. What’s attractive is that its formula contains bond order information, thus it can generate structures with reasonable coordination numbers.

\[ U_T = \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1, j \neq i}^{N} f_c (r_{ij}) (U_R (r_{ij}) + b_{ij} U_A (r_{ij})) \]  \hspace{1cm} (3.1.17)

where:

\[ f_c (r_{ij}) = \begin{cases} 
1 & r_{ij} < R_{ij} \\
\frac{1}{2} + \frac{1}{2} \cos \frac{r_{ij} - R_{ij}}{S_{ij} - R_{ij}} & R_{ij} < r_{ij} < S_{ij} \\
0 & S_{ij} < r_{ij} \end{cases} \]  \hspace{1cm} (3.1.18)

\[ U_R (r_{ij}) = A_{ij} \exp (-\lambda_{ij} r_{ij}) \]  \hspace{1cm} (3.1.19)

\[ U_A (r_{ij}) = -B_{ij} \exp (-\mu_{ij} r_{ij}) \]  \hspace{1cm} (3.1.20)

\[ b_{ij} = \chi_{ij} \left( 1 + \beta_{ij} \epsilon_{ij} \right)^{-\frac{1}{\gamma_{ij}}} \]  \hspace{1cm} (3.1.21)

\[ \xi_{ij} = \sum_{k=1}^{N} f_c (r_{ik}) \omega_{ik} g_{ijk} \]  \hspace{1cm} (3.1.22)

\[ g_{ijk} = 1 + \left( \frac{c_i}{d_i} \right)^2 - \frac{c_i^2}{d_i^2 + (h_i - \cos \theta_{ijk})^2} \]  \hspace{1cm} (3.1.23)

In (3.1.23), \(\theta_{ijk}\) is the angle formed by three atoms \(i, j\) and \(k\), atom \(i\) being the center.

12. Extended Lennard-Jones potential (ELJ\(_N\)). This is a more accurate version of Lennard-Jones potential, although at small size its GM does not change much compared to that of a usual LJ cluster.

\[ U_{ELJ} = \sum_{i=1}^{N} \sum_{i<j}^{N} \sum_{k=6}^{k_{\text{max}}} \frac{c_k}{r_{ij}^{6/k}} \]  \hspace{1cm} (3.1.24)

Now we will see how to perform global optimization on these clusters.
3.2 Example 1: 38 Lennard-Jones Particles

We have 38 particles interacting with Lennard-Jones potential (3.1.2), denoting as LJ\(_{38}\). We want to get its global minimum.

Step 1: In the ABCluster distribution, go to the directory testfiles/atomiccluster.

Step 2: Use abcinp to generate the input files:

```
../../abcinp lj38 1 LennardJones 5.0 30 300 5 30 38 C
```

Parameters for atom 0: sigma epsilon > 1.4 1.0

This line will be explained in the next section.

Step 3: Run the global optimization:

```
../../atom-optimizer lj38.inp > lj38.out
```

Below is everything you get after optimization:

- **lj38.out** The main output file.
- **lj38.xyz** The global minimum in XYZ format. It can be read by e.g. VMD, CYLView.
- **lj38.gjf** The global minimum in Gaussian input format. It can be read by e.g. GaussView.
- **lj38-LM** Contains the local minima, each one having two files in XYZ and Gaussian input format, respectively. They are sorted by energy-increasing order, e.g. 0.xyz is lower in energy than 13.xyz.
- **abcluster*.xyz/gjf** The file containing the current found global minimum during the running of rigidmol-optimizer. If rigidmol-optimizer crashes before normal termination, one can use this abcluster*.xyz to start a new optimization.

Now, you can check the final global minimal energy in lj38.out:

```
29  -170.78466518
* Final Global Minimal Energy:  -173.92842659
* The Global Minimum is saved as: [ lj38.(gjf/./xyz) ]
```

with the geometry in lj38.xyz (See Figure 3.2.1 ). Also, in the directory lj38-LM, there are 30 local minima for you.

3.3 About abcinp

Now we want to explain in detail how to generate the input files with abcinp. abcinp will print help information when run without any arguments:

```
abcinp filename kind_of_atoms forcefield amplitude sn gmax glimit nsaves natoms1 symbol1 ...
-- For atomic clusters:
forcefield = LennardJones: sigma epsilon
forcefield = CoulombLennardJones: q sigma epsilon
forcefield = Morse: power epsilon beta r0
forcefield = CoulombBornMayer: q B rho
forcefield = CoulombMorseRepulsion: q D alpha rho C
forcefield = Girifalco: nC A B d
forcefield = Dzugutov: a alpha kF b sigma m rC V0
```
3.3. ABOUT ABCINP

forcefield = Gupta: \quad A \times i d \ p \ q
forcefield = SuttonChen: \quad p \ q \ \epsilon \ sion \ a \ c
forcefield = SuttonChenMJB: \quad p \ \epsilon \ sion \ a \ c \ \alpha
forcefield = Tersoff: \quad A \ B \ \lambda \ mba \ m \ c \ d \ h \ R \ S \ \chi \ ome
forcefield = ExLennardJones: \quad maxk \ c6 \ c7 \ldots \ cmaxk

We will interpret all the arguments:

- filename The unique filename. The generated input files will be named as filename.inp, filename.par and filenamei.xyz.
- kind_of_atoms The number of atomic types. For example, in the case of MgO clusters, kind_of_atoms = 2.
- forcefield The name of the force field. Available choices for atomic clusters are:
  - LennardJones, see (3.1.2).
  - CoulombLennardJones, see (3.1.3).
  - Morse, see (3.1.4).
  - CoulombBornMayer, see (3.1.1).
  - CoulombMorseRepulsion, see (3.1.5).
  - Girifalco, see (3.1.7).
  - Dzugutov, see (3.1.6).
  - Gupta, see (3.1.11) and (3.1.12).
  - SuttonChen, see (3.1.13) and (3.1.14).
  - SuttonChenMJB, see (3.1.15) and (3.1.16).
  - Tersoff, see (3.1.17).
  - ExLennardJones, see (3.1.24)

For “special particle clusters”, see Chapter 6.

- amplitude The estimated size of the cluster in Å, corresponding to “L” mentioned in Section 2.
- sn The size of the population of trial solutions, corresponding to “SN” mentioned in Section 2. Recommended value: 10 to 300.
CHAPTER 3. ATOMIC CLUSTERS WITH FORCE FIELDS

• \texttt{gmax} The maximum cycle number, corresponding to “gmax” mentioned in Section 2. Recommended value: 100 to 100000 or larger.

• \texttt{glimit} The scout limit, corresponding to “glimit” mentioned in Section 2. Recommended value: 2 to 10.

• \texttt{nasves} The number of local minima to be saved.

• \texttt{natoms1 symbol1 ...} For all kinds of atoms, its number and symbol must be given. Here the symbol is not necessarily an element symbol like C or Mg, but can be any strings, like CT2.

After this, the program will require you to input force field parameters for different pairs of atoms. The parameters are exactly the ones used from (3.1.1) to (3.1.14) in Section 6.6.

Now we give explicit correspondence between the variables in the program input and the potential expression:

Lennard-Jones, (3.1.2) \quad \text{sigma - } \sigma; \text{ epsilon - } \epsilon
Coulomb–Lennard-Jones, (3.1.3) \quad q - q; \text{ sigma - } \sigma; \text{ epsilon - } \epsilon
Morse, (3.1.4) \quad \text{power - } n; \text{ epsilon - } \epsilon; \text{ beta - } \beta; \text{ r0 - } r^0
Coulomb–Morse–Repulsion, (3.1.5). \quad q - q; \text{ D - D}; \text{ alpha - } \alpha; \text{ rho - } \rho; \text{ C - C}
Girifalco, (3.1.6) \quad a - a; \text{ alpha - } \alpha; \text{ kF - } k_F; \text{ b - b}; \text{ sigma - } \sigma; \text{ m - m}; \text{ rC - } r_C; \text{ V0 - V}_0
Gupta, (3.1.11) \quad A - A; \text{ xi - } \xi; \text{ d - d}; \text{ p - p}; \text{ q - q}
Sutton–Chen, (3.1.13) \quad p - p; \text{ q - q}; \text{ epsilon - } \epsilon; \text{ a - a}; \text{ c - c}
Modified Sutton-Chen (by Januszko–Bose), (3.1.15) \quad p - p; \text{ epsilon - } \epsilon; \text{ a - a}; \text{ c - c}; \text{ alpha - } \alpha
Tersoff, (3.1.17) \quad A - A; \text{ B - B}; \text{ lambda - } \lambda; \text{ mu - } \mu; \text{ beta - } \beta; \text{ m - m}; \text{ c - c}; \text{ d - d}; \text{ h - h}; \text{ R - R}; \text{ S - S}; \text{ chi - } \chi; \text{ omega - } \omega
Extended Lennard-Jones, (3.1.24) \quad \text{maxk - } k_{\text{max}}; \text{ c6 - } c_6, \ldots

For instance, in the last section, we use the following command:

```
../../abcinp lj38 1 LennardJones 5.0 30 300 5 30 38 C
```

Parameters for atom 0: sigma epsilon > 1.4 1.0

This means: Perform a global optimization on a LennardJones cluster, with \( L = 5.0, SN = 30, g_{\text{max}} = 300, g_{\text{limit}} = 5, 30 \) local minima will be saved. We have 1 kind of particles, its number, and symbol is 38 and C, respectively. The generated file will be named as lj38*.

After this, you will get three files:

• \texttt{lj38.inp} The main input file.

• \texttt{lj38.par} The parameter file.

• \texttt{lj38i.xyz} Initial guess in XYZ format.

Open \texttt{lj38.inp} you will find this:

```
38 # number of atoms
lj38i.xyz # initial guess file; * - random guess
LennardJones # initial field types
lj38.par # force field parameters
```
3.4. Example 2: \((\text{Mg}^Q\text{O}^{Q^-})_{20}\)

In this section, we will find that the parameters can change the global minima significantly. We consider \((\text{MgO})_{20}\). It can be described by Coulomb–Born–Mayer potential 3.1.1. A set of parameters is\([17]\):

\[
\begin{align*}
B(\text{Mg} - \text{O}) &= 821.6; \\
B(\text{Mg} - \text{Mg}) &= 0.0; \\
B(\text{O} - \text{O}) &= 22764 \\
\rho(\text{Mg} - \text{O}) &= 0.3242; \\
\rho(\text{Mg} - \text{Mg}) &= 0.1; \\
\rho(\text{O} - \text{O}) &= 0.1490 \\
g_{\text{Mg}} &= +2.0; \\
g_{\text{O}} &= -2.0
\end{align*}
\]

If we use \(L = 5.0\), \(SN = 100\), \(g_{\text{max}} = 100\), \(g_{\text{limit}} = 5\), then we can perform the global optimization:

```bash
../../abcinp mg20o20-q2 2 CoulombBornMayer 5.0 100 100 5 30 20 Mg 20 O
```

Parameters for atom 0: \(q > +2\)
Parameters for atom 1: \(q > -2\)
Parameters for atom-pair 0-0: \(B \rho > 0.0 0.1\)
Parameters for atom-pair 0-1: \(B \rho > 821.6 0.3242\)
Parameters for atom-pair 1-1: \(B \rho > 22746 0.1490\)

Since we have 2 kinds of atoms, “Mg” and “O”, we have to write \(20 \text{Mg} 20 \text{O}\) to indicate that we have 20 Mg and 20 O atoms. Now perform the optimization:

```bash
atom-optimizer mg20o20-q2.inp > mg20o20-q2.out
```

Before looking at its geometry, what will it be if the formal charge \(Q\) on Mg and O are not 2.0, but other numbers, e.g. 1.5 and 1.0? Let’s try it. Generate the input files for \(\text{Mg}^{1.5+}\text{O}^{1.5-}\)

```bash
../../abcinp mg20o20-q1.5 2 CoulombBornMayer 5.0 100 100 5 30 20 Mg 20 O
```

Parameters for atom 0: \(q > +1.5\)
Parameters for atom 1: \(q > -1.5\)
Parameters for atom-pair 0-0: \(B \rho > 0.0 0.1\)
Parameters for atom-pair 0-1: \(B \rho > 821.6 0.3242\)
Parameters for atom-pair 1-1: \(B \rho > 22746 0.1490\)

and for \(\text{Mg}^{1.0+}\text{O}^{1.0-}\)

```bash
../../abcinp mg20o20-q1 2 CoulombBornMayer 5.0 100 100 5 30 20 Mg 20 O
```

Parameters for atom 0: \(q > +1.0\)
Parameters for atom 1: \(q > -1.0\)
Parameters for atom-pair 0-0: \(B \rho > 0.0 0.1\)
Parameters for atom-pair 0-1: \(B \rho > 821.6 0.3242\)
Parameters for atom-pair 1-1: \(B \rho > 22746 0.1490\)
Run them!

atom-optimizer mg20o20-q1.5.inp > mg20o20-q1.5.out
atom-optimizer mg20o20-q1.inp > mg20o20-q1.out

Now check mg20o20-q2.xyz, mg20o20-q1.5.xyz and mg20o20-q1.xyz, you will observe that the global minimum is a sphere, tube and cuboid, respectively!

Figure 3.4.1: The global minimum of (MgO)$_{20}$.

3.5 Example 3: Silver and Copper Clusters

3.5.1 Ag$_{38}$ and Cu$_{38}$

Assume you get a silver and copper cluster. By mass spectroscopy, you know that their formulae are Ag$_{38}$ and Cu$_{38}$, respectively. What are their most stable structures?

The best potential for metal clusters is Gupta potential (3.1.11). From Reference[18], one can find their Gupta potential parameters:

\[
\begin{array}{cccccc}
A & \xi & d & p & q \\
Ag-Ag & 0.1028 & 1.1780 & 2.8885 & 10.928 & 3.1390 \\
Cu-Cu & 0.0855 & 1.2240 & 2.5562 & 10.960 & 2.2780 \\
Cu-Ag & 0.0980 & 1.2274 & 2.7224 & 10.700 & 2.8050 \\
\end{array}
\]

OK, now we can perform the global optimization. Let’s see Ag$_{38}$ first.

Step 1: Use abcinp to generate the input files:

```
abcinp Ag38 1 Gupta 10 100 100 5 20 38 Ag
```

Parameters for atom-pair 0-0: A xi d p q > 0.1028 1.1780 2.8885 10.928 3.1390

Step 2: Run the optimization:

```
nohup atom-optimizer Ag38.inp > Ag38.out &
```

Step 3: Now we examine the structure by opening the file Ag38.xyz. See Figure 3.5.1.

Now we do the same for Cu$_{38}$.

Step 1: Use abcinp to generate the input files:

```
abcinp Cu38 1 Gupta 10 100 100 5 20 38 Cu
```

Parameters for atom-pair 0-0: A xi d p q > 0.0855 1.2240 2.5562 10.960 2.2780
3.6. EXAMPLE 4: PEROVSKITE CLUSTERS (CATIO$_3)_5$

Step 2: Run the optimization:

```
nohup atom-optimizer Cu38.inp > Cu38.out &
```

Step 3: Now we examine the structure by opening the file Ag38.xyz. See Figure 3.5.1.

Interestingly, both are a truncated octahedron.

![Ag$_{38}$ and Cu$_{38}$](image)

Figure 3.5.1: The global minimum of Ag$_{38}$ and Cu$_{38}$.

3.5.2 Ag$_{32}$Cu$_6$

However, you may get a “nanoalloy”, a mixture of silver and copper atoms. For example, the alloy is Ag$_{32}$Cu$_6$. What shape will it be? Still truncated octahedron? Let’s see.

Step 1: Use abcinp to generate the input files:

```
abcinp Ag32Cu6 2 Gupta 10 3000 5000 5 20 32 Ag 6 Cu
Parameters for atom-pair 0-0: A xi d p q > 0.1028 1.1780 2.8885 10.928 3.1390
Parameters for atom-pair 0-1: A xi d p q > 0.0980 1.2274 2.7224 10.700 2.8050
Parameters for atom-pair 1-1: A xi d p q > 0.0855 1.2240 2.5562 10.960 2.2780
```

Note that, for mixed alloys, the number of steps required to find the global minimum may be very large. In ABCluster 1.1 or 1.2 we have to set very large $SN$ and $g_{\text{max}}$, see examples files in Ag32Cu6-slow-1.1 where $SN = 3000$ and $g_{\text{max}} = 5000!$. Fortunately, from ABCluster 1.3, we improved the algorithm significantly and now moderately parameters work quiet well! Try to set $SN = 300$ and $g_{\text{max}} = 300$. During the running, you can examine the current global minimum by examining the temporal file abcluster*.xyz.

Step 2: Run the optimization:

```
nohup atom-optimizer Ag32Cu6.inp > Ag32Cu6.out &
```

Step 3: Now we examine the structure by opening the file Ag32Cu6.xyz. See Figure 3.5.2. Amazingly, it is a “pancake”, with the six copper atoms forming a hexagonal ring!

3.6 Example 4: Perovskite Clusters (CaTiO$_3)_5$

Perovskite recently attracts much attention from its scientific community due to its unexpected application in solar cells. Assume now you want to know the structure of (CaTiO$_3)_5$, you can do this job by ABCluster. For such systems, the so-called Coulomb–Morse–Repulsion potential (3.1.5) can be used. One can get the interaction parameter Ca-O, Ti-O and O-O from Ref. [8]. For cation interaction Ca-Ca and Ti-Ti, only the Coulomb term ($1/r$) and strong repulsion term ($1/r^{12}$) are considered (Assume $C = 22$), the Morse term can be assumed to be zero (thus $D$, $\alpha$ and $\rho$ are all zero). The final parameters are:
CHAPTER 3. ATOMIC CLUSTERS WITH FORCE FIELDS

Figure 3.5.2: The global minimum of Ag\textsubscript{32}Cu\textsubscript{6}.

<table>
<thead>
<tr>
<th></th>
<th>q(Ca) = +1.2</th>
<th>q(Ti) = +2.4</th>
<th>q(O) = −1.2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ca-Ca</td>
<td>D</td>
<td>α</td>
<td>ρ</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Ca-Ti</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Ca-O</td>
<td>0.030211</td>
<td>2.241334</td>
<td>2.923245</td>
</tr>
<tr>
<td>Ti-Ti</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Ti-O</td>
<td>0.024235</td>
<td>2.254703</td>
<td>2.708943</td>
</tr>
<tr>
<td>O-O</td>
<td>0.042395</td>
<td>1.379316</td>
<td>3.618701</td>
</tr>
</tbody>
</table>

Now the optimization starts.

Step 1: Use `abcinp` to generate the input files:

```
abcinp catio3 3 CoulombMorseRepulsion 10 100 1000 5 10 5 Ca 5 Ti 15 0
Parameters for atom 0: q > +1.2
Parameters for atom 1: q > +2.4
Parameters for atom 2: q > −1.2
Parameters for atom-pair 0-0: D alpha rho C > 0 0 0 22.0
Parameters for atom-pair 0-1: D alpha rho C > 0 0 0 22.0
Parameters for atom-pair 0-2: D alpha rho C > 0.030211 2.241334 2.923245 5.0
Parameters for atom-pair 1-1: D alpha rho C > 0 0 0 22.0
Parameters for atom-pair 1-2: D alpha rho C > 0.024235 2.254703 2.708943 1.0
Parameters for atom-pair 2-2: D alpha rho C > 0.042395 1.379316 3.618701 22.0
```

Step 2: Run the optimization:

```
nohup atom-optimizer catio3.inp > catio3.out &
```

Step 3: Now we examine the structure by opening the file `catio3.xyz`. See Figure 3.6.1. Note that it is very different from a bulk, perovskite crystal structure.

### 3.7 Example 5: C\textsubscript{16} and Ge\textsubscript{16} Clusters

C\textsubscript{16} and Ge\textsubscript{16} clusters are inorganic clusters with very delocalized interactions. A suitable description requires quantum chemistry. However one may use Tersoff potential for a first exploration. The Tersoff potential parameters for C and Ge are listed below:
3.7. EXAMPLE 5: \( C_{16} \) AND GE\(_{16} \) CLUSTERS

![Figure 3.6.1: The global minimum of \((\text{CaTiO}_3)_5\).](image)

Using these parameters, it is not difficult to get the global minima of \( C_{16} \) and \( \text{Ge}_{16} \). For \( C_{16} \):

Step 1: Use `abcinp` to generate the input files:

```
abcinp c 1 Tersoff 5 20 2000 5 10 $1 c
```

Parameters for atom 0: \( A \ B \ \lambda \ \mu \ \beta \ m \ c \ d \ h \ R \ S >\)

\[
\begin{array}{cccc}
A & 1393.6 & 1769.0 \\
B & 346.7 & 419.23 \\
\lambda & 3.4879 & 2.4451 \\
\mu & 2.2119 & 1.7047 \\
\beta & 0.0000001524 & 0.00000090166 \\
m & 0.72751 & 0.75627 \\
c & 38049.0 & 106430.0 \\
d & 4.384 & 15.652 \\
h & -0.57058 & -0.43884 \\
R & 1.8 & 2.8 \\
S & 2.1 & 3.1 \\
\omega & 1.0 & 1.0 \\
\xi & 1.0 & 1.0 \\
\end{array}
\]

Parameters for atom-pair 0-0: \( k\alpha \ \omega \ >\)

\[
\begin{array}{cccc}
1.0 & 1.0 \\
\end{array}
\]

Step 2: Run the optimization:

```
nohup atom-optimizer c.inp > c.out &
```

For \( \text{Ge}_{16} \) you can do the same thing. The global minima are shown in Figure 3.7.1.
Figure 3.7.1: The global minimum of (C)$_{16}$ and (Ge)$_{16}$. 
Chapter 4

Rigid Molecular Clusters with Force Fields

4.1 Introduction

Molecular clusters are very important in several field of chemistry, biology, and physics. In the current version of ABCluster, the molecules in a cluster are assumed to be rigid, meaning that the internal degrees of freedoms (bond lengths, bond angles, dihedral angles, etc.) are kept unchanged during the optimization. For small molecules, this is a very good approximation. Thus each molecule in the cluster can be described by six external degrees of freedoms: the coordinates of its geometrical center $\mathbf{R} \equiv \{X, Y, Z\}$ and three Euler angles $\Omega \equiv \{\alpha, \beta, \gamma\}$ relative to its pre-defined body-fixed coordinate system.

The potential energy of the rigid molecular cluster is assumed to be of the following form:

$$U(Q) = \sum_{I=1}^{N} \sum_{i \in I} \left( \frac{e^2}{4\pi\varepsilon_0} \frac{q_i q_j}{r_{ij}} + 4\epsilon \left( \left( \frac{\sigma_{ij}}{r_{ij}} \right)^{12} - \left( \frac{\sigma_{ij}}{r_{ij}} \right)^6 \right) \right) + \sum_{I=1}^{N} \sum_{i \in I} (q_i e F \cdot \mathbf{z}_i)$$

(4.1.1)

Here, (4.1.1) describes the intermolecular Coulomb and Lennard-Jones interactions. This simple form enables one to compute large systems fast and quickly gain chemical insights. Also, from ABCluster 2.0, an additional external electric field energy term is added, with $F$ being the electric field strength, thus one can study the clusters under the influence of external electric field. One can get a lot of local minima by ABCluster first, and then use high-level theory, like quantum chemistry, to further study these isomers.

The unit of $\epsilon$, $\sigma$ and $F$ in (4.1.1) is kJ mol$^{-1}$, Å, and VÅ$^{-1}$, respectively. These parameters can be obtained from modern force fields, like CHARMM[19], OPLS[20] and AMBER[21]. Some molecule files suitable for ABCluster are available in the distribution in

```
testfiles/rigidmolecularcluster/charmm36

```

testfiles/rigidmolecularcluster/oplsaa

Also, the latest force field parameters will be available on the ABCluster website.
An important feature appeared from ABCluster 1.5 is that molecular clusters can be set in 0D (isolate), 1D (wire), 2D (surface) and 3D (crystal) periodic environment. Thus “in principle” one can predict inorganic and organic molecular crystal structures. However, the accurate crystal structure prediction (CSP) is still a very challenging problem and very highly accurate methods have to be applied: Sometimes an energy difference of 0.1 kJ mol\(^{-1}\) can lead to a different crystal structure. Therefore, the crystal structures predicted by ABCluster can be used as initial guesses but not so suitable as the final results. The author is working hard to make an accurate and cheap prediction possible in future versions of ABCluster.

### 4.2 Example 1: (H\(_2\)O)\(_6\)

We want to know the most stable structure of (H\(_2\)O)\(_6\). We describe water by TIP4P model. This can be done in the following steps.

Step 1: In the ABCluster distribution, go to the directory testfiles/rigidmolecularcluster. Note that there is a directory charmm36.

Step 2: Prepare an input file named h2o6.inp with the following content:

```
input . cluster # cluster file name
20 # population size
20 # maximal generations
3 # scout limit
4.00000000 # amplitude
h2o6 # save optimized configuration
30 # number of LMs to be saved
```

This file is self-explained:

- Line 1: The name of the file that contains the cluster components and geometry, it will be explained in the next step.

- Line 2: The size of the population of trial solutions, corresponding to “SN” mentioned in Section 2. The larger the cluster, the larger SN is required.

- Line 3: The maximum cycle number, corresponding to “g\(_{\text{max}}\)” mentioned in Section 2. For such small systems, 20 is sufficient. For larger ones, perhaps 100000 is required!

- Line 4: The scout limit, corresponding to “g\(_{\text{limit}}\)” mentioned in Section 2. Usually 3 to 10 is OK.

- Line 5: The estimated size of the cluster in Å, corresponding to “L” mentioned in Section 2. For an efficient optimization of large systems, this value should be 1.5 to 3 times of the system size.

- Line 6: The name to save the results.

- Line 7: The number of local minima to be saved.

Step 3: Prepare the cluster file named input.cluster with the following content:

```
1
charmm36/tip4p.xyz 6
* 4.0000
```

Here:
4.2. EXAMPLE 1: \((H_2O)_6\)

- Line 1: The number of different components in the cluster. For \((H_2O)_6\) this is 1. For \(Na^+(H_2O)_3(CH_3OH)_2\), this is 3.

- Line 2: The file name (with path!) containing the coordinates and force field parameters, and the number of this molecules in the cluster. If there are more than one components in the cluster, then each component should be described by a line like this.

- Line 3: Here, *4.0000 means generating a random initial guess with \(L = 4.0000\). It can also have another form:

```
1
charmm36/tip4p.xyz 6
Water hexamer
3.5 7.0 -0.2 1.3 2.9 0.6
4.0 0.9 3.1 4.3 3.2 3.8
-1.8 4.4 1.0 0.8 0.2 0.6
4.2 3.4 3.9 1.9 0.2 3.2
3.6 4.7 2.9 3.6 1.7 1.7
5.1 5.2 -0.6 1.6 2.8 3.5
```

In ABCluster, as long as the first line after cluster components is not of the form like *5.000* but anything else (e.g. a title), then in the following one has to give an explicit initial guess in the form \(X, Y, Z, \alpha, \beta\) and \(\gamma\).

Step 4: Now we have h2o6.inp and input.cluster, we can run the global optimization:

```
../../rigidmol-optimizer h2o6.inp > h2o6.out
```

When the optimization is finished, we get the following files.

- **h2o6.out** The main output file.
- **h2o6-OPT.xyz** The global minimum in XYZ format. It can be read by e.g. VMD, CYLView.
- **h2o6-OPT.gjf** The global minimum in Gaussian input format. It can be read by e.g. GaussView.
- **h2o6-OPT.cluster** The global minimum in ABCluster format. It can only be read by rigidmol-optimizer. Also, it can be used as the initial guess of a new optimization.
- **h2o6-LM** Contains the local minima, each one having three files in XYZ, Gaussian input, and ABCluster format. They are sorted by energy-increasing order, e.g. 0.xyz is lower in energy than 13.xyz.
- **abcluster*.xyz/gjf/cluster** The file containing the current found global minimum during the running of rigidmol-optimizer. If rigidmol-optimizer crashes before normal termination, one can use this *.cluster to start a new optimization.

Now let us see the result. The h2o6.out is similar to that generated by atom-optimizer, thus will not be explained again here. Open h2o6-OPT.xyz, we found that the global minimum is the cage isomer. Open h2o6-OPT.xyz, we found that the global minimum is the cage isomer. Open h2o6-LM/2.xyz, we found a local minimum: the prism isomer. See Figure 4.2.1. To get a reliable conclusion on which one is more stable, please use high-level methods like quantum chemistry for further study. Of course, we can play with all 30 local minima!
4.3 Example 2: Li\(^+\), Na\(^+\) and Cs\(^+\) in (C\(_6\)H\(_6\))\(_6\)

The section title gives three cation-\(\pi\) systems. Let’s see what their global minima look like.

Step 1: In the ABCluster distribution, go to the directory testfiles/rigidmolecularcluster. Note that there is a directory charmm36.

Step 2: Optimize Li\(^+\)(C\(_6\)H\(_6\))\(_6\). Prepare two files: li-ben.inp and li-ben.cluster:

```plaintext
li-ben.cluster  # cluster file name
20 # population size
20 # maximal generations
3 # scout limit
10.00000000 # amplitude
li-ben # save optimized configuration
30 # number of LMs to be saved
```

<table>
<thead>
<tr>
<th>charmm36/c6h6.xyz 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>charmm36/li.xyz 1</td>
</tr>
<tr>
<td>* 10.0000</td>
</tr>
</tbody>
</table>

Then run the optimization:

```
../../rigidmol-optimizer li-ben.inp > li-ben.out
```

Step 3: Do the similar thing for Na\(^+\)(C\(_6\)H\(_6\))\(_6\) and Cs\(^+\)(C\(_6\)H\(_6\))\(_6\). Remember to change the file name from li* to na* or cs*, and in *.cluster, change charmm36/li.xyz to charmm36/na.xyz or charmm36/cs.xyz.

Now let’s open li-ben-OPT.xyz, na-ben-OPT.xyz and cs-ben-OPT.xyz to see their global minima, shown in Figure 4.3.1.

One can see that in the first solvation shell, Li\(^+\) has only 2 C\(_6\)H\(_6\), but Na\(^+\) and Cs\(^+\) have 5. Moreover, the remaining C\(_6\)H\(_6\) is closer to Cs\(^+\) than that to Na\(^+\). This is obviously related to their charge density. Maybe you want to see what will happen for K\(^+\) or SO\(_4^{2-}\). Just try it! ABCluster can do many amazing things!

4.4 Example 3: (CH\(_2\)=CH\(_2\))\(_{13}\) and Build New Force Fields

Ethene is of fundamental importance in chemical industry. Now we want to know the global minimum of (CH\(_2\)=CH\(_2\))\(_{13}\). However, it seems that there is no ethene in
4.4. EXAMPLE 3: \((\text{CH}_2=\text{CH})_6\) AND BUILD NEW FORCE FIELDS

Figure 4.3.1: The global minima of Li\(^+\)(C\(_6\)H\(_6\))\(_6\), Na\(^+\)(C\(_6\)H\(_6\))\(_6\) and Cs\(^+\)(C\(_6\)H\(_6\))\(_6\).

testfiles/rigidmolecularcluster/charmm36. Therefore we have to build it by ourselves.

Step 1: Construct the geometry of \(\text{CH}_2=\text{CH}_2\). An optimization at B3LYP/6-31G(d) level is sufficient for our purpose. The final geometry is stored in a file called c2h4.xyz in XYZ format:

| ethene | C 0.00000000 0.00000000 0.66542300 |
| C     | 0.00000000 0.92366200 1.23955100  |
| H     | 0.00000000 -0.92366200 1.23955100  |
| H     | 0.00000000 0.00000000 -0.66542300  |
| C     | 0.00000000 -0.92366200 -1.23955100 |
| H     | 0.00000000 0.92366200 -1.23955100  |

Step 2: Construct the force field parameters of \(\text{CH}_2=\text{CH}_2\). Since in this manual, we always use CHARMM force field, for consistency we also use this for ethene. You can get the lastest CHARMM force field files from:

http://mackerell.umaryland.edu/charmm_ff.shtml

At the time this manual is written, the latest version is CHARMM36. After you download toppar_c36_aug14.tgz and unzip it, fortunately, there are parameters for ethene.

In the file toppar/top_all36_cgenff.rtf you find the following words:

| RESI ETHE | 0.00 ! C2H4 ethylene, yin/adm jr. |
| GROUP    |                               |
| ATOM C1 CG2D2 -0.42 ! |
| ATOM H11 HGA5 0.21 ! H11 H21 |
| ATOM H12 HGA5 0.21 ! / / |
| GROUP    | ! C1=C2 |
| ATOM C2 CG2D2 -0.42 ! / / |
| ATOM H21 HGA5 0.21 ! H12 H22 |
| ATOM H22 HGA5 0.21 ! |
| BOND C1 H11 C1 H12 |
Take the first line ATOM C1 CG2D2 -0.42 as an example, C1 indicates the corresponding atom; CG2D2 is the atomic type; -0.42 is its q parameter in (4.1.1).

To find its ϵ and σ parameters, we check the file toppar/par_all36.cgenff.rtf, and find this:

<table>
<thead>
<tr>
<th>Atomic Type</th>
<th>q (kcal/mol)</th>
<th>ϵ (kcal/mol)</th>
<th>σ (Å)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CG2D2</td>
<td>-0.0640</td>
<td>2.0800</td>
<td></td>
</tr>
</tbody>
</table>

-0.0640 and 2.0800 are ϵ and σ, respectively. However, since CHARMM and ABCluster use different formulae, we have to transform them! The transformation formulae are:

\[
\epsilon_{\text{ABCluster}} = \epsilon_{\text{CHARMM}} \times (-4.184)
\]

Thus its final value is:

\[
q = -0.42
\]

\[
\epsilon = -0.0640 \times (-4.184) = 0.2678
\]

\[
\sigma = 2.0800 \times 2^{\frac{3}{4}} = 3.7061
\]

The parameters of H can be found in a similar way:

\[
q = +0.21
\]

\[
\epsilon = -0.0260 \times (-4.184) = 0.1088
\]

\[
\sigma = 1.2600 \times 2^{\frac{3}{4}} = 2.2451
\]

Thus the final parameter file c2h4.xyz for CH₂=CH₂ is:

<table>
<thead>
<tr>
<th>q (kcal/mol)</th>
<th>ϵ (kcal/mol)</th>
<th>σ (Å)</th>
</tr>
</thead>
<tbody>
<tr>
<td>-0.42</td>
<td>0.2678</td>
<td>3.7061</td>
</tr>
<tr>
<td>+0.21</td>
<td>0.1088</td>
<td>2.2451</td>
</tr>
<tr>
<td>-0.42</td>
<td>0.2678</td>
<td>3.7061</td>
</tr>
<tr>
<td>+0.21</td>
<td>0.1088</td>
<td>2.2451</td>
</tr>
<tr>
<td>+0.21</td>
<td>0.1088</td>
<td>2.2451</td>
</tr>
</tbody>
</table>

Note that there is a comment line between the geometry and parameter section. The words after # are comments.

Step 3: Now prepare the files c2h413.inp and input.cluster:
4.5. **Example 4: La$^{3+}$(H$_2$O)$_{15}$ at Gold {100} Surface**

Based on the previous sections, I think it will not be difficult for you to find the GM of La$^{3+}$(H$_2$O)$_{15}$. Now we want to do better: find its GM at the surface of, say gold. To perform the global optimization of La$^{3+}$(H$_2$O)$_{15}$ at gold {100} surface, prepare a file `mol.cluster` with the following contents:

```
3
../charmm36/1a.xyz 1
../charmm36/au-surf-100-5x5.xyz 1
../charmm36/tip4p.xyz 15
* 20
```
CHAPTER 4. RIGID MOLECULAR CLUSTERS WITH FORCE FIELDS

Figure 4.4.1: The global minimum of (CH₂=CH₂)₁₃.

Here, au-surf-100-5x5.xyz is a file containing 25 atoms at the {100} surface of gold crystal and the corresponding parameters. Then prepare the input file mol.inp:

```
mol.cluster # cluster file name
30 # population size
100 # maximal generations
3 # scout limit
4.00000000 # amplitude
mol # save optimized configuration to .xyz and .gjf
.
gjf
30 # number of LMs to be saved
```

Now run the optimization:

```
../../rigidmol-optimizer mol.inp > mol.out
```

After a while, you can find the GM from the file mol-OPT.xyz or mol-OPT.gjf. It can be found in Figure 4.5.1. We find that La³⁺ has a water coordinate number of 9, with monocapped square antiprism geometry. Perhaps you know that in bulk water, La³⁺(H₂O)₉ is of tricapped trigonal prism geometry! (Confirm this by a global optimization of, say La³⁺(H₂O)₁₅.)

If you want to do similar calculations on other surfaces, you can first construct a surface from the CIF file of the metal, cut the surface you want (e.g., {100}, {111}, {532}), take the coordinates of, e.g. 5 × 5 = 25 atoms and make a parameter file as Section 4.4 instructed. For parameters, please refer to J. Phys. Chem. C, 2008, 112, 17281[22].

4.6 Example 5: (H₂O)₂(CH₃OH)₂ in 1D Environments

I think now you can easily use ABCluster to find the GM of an isolate (H₂O)₂(CH₃OH)₂, with the following files h2och3oh.inp and h2och3oh.cluster:

```
h2och3oh.cluster # cluster file name
20 # population size
50 # maximal generations
```
Figure 4.5.1: The global minimum of La$^{3+}$(H$_2$O)$_{15}$ at the gold \{100\} surface and in bulk water.

<table>
<thead>
<tr>
<th>3</th>
<th># scout limit</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.00000000</td>
<td># amplitude</td>
</tr>
<tr>
<td>h2och3oh</td>
<td># save optimized configuration to .xyz and .gjf</td>
</tr>
<tr>
<td>30</td>
<td># number of LMs to be saved</td>
</tr>
</tbody>
</table>

What will (H$_2$O)$_2$(CH$_3$OH)$_2$ look like, if it is not in gas (isolate) phase but in periodic environments? From ABCluster 1.5 we can do this! ABCluster can optimize both the cluster structure and the lattice parameters a, b and c. First, let’s try to see what a 1D-(H$_2$O)$_2$(CH$_3$OH)$_2$ looks like. In order to do this, you only need to change h2och3oh.cluster, change it like this:

\[ 2 \]
\[ .../charm36/tip4p.xyz 2 \]
\[ .../charm36/ch3oh.xyz 2 \]
\[ * 3 \]
\[ 1 \]
\[ 1.0 0.0 0.0 \]

where the additional 1 means that we consider a 1D periodic system. The following 1.0 0.0 0.0 is a guess translational vector a (must be provided, but ABCluster will optimize it). Now perform the optimization with rigidmol-optimizer. The results are stored in h2och3oh-0PT.* as well as in h2och3oh-0PT-LM. Note that, in these files, the last few lines containing TV ... are a. The 1D infinite chain of (H$_2$O)$_2$(CH$_3$OH)$_2$ are shown in Figure 4.7.1. Obviously, this is an infinite chain held by hydrogen bonds, and perhaps the methyl group packing!

### 4.7 Example 6: HNO$_3$(H$_2$O)$_{10}$ in an Electric Field

Now you can easily perform a global optimization of the cluster HNO$_3$(H$_2$O)$_{10}$ For example, you can prepare the input file ef-0.inp and ef-0.cluster as below, and store the local minima in ef-0-LM.

| ef-0.cluster # cluster file name |
CHAPTER 4. RIGID MOLECULAR CLUSTERS WITH FORCE FIELDS

Figure 4.6.1: The GM of \((\text{H}_2\text{O})_2(\text{CH}_3\text{OH})_2\) in 1D environments.

```
30          # population size
50          # maximal generations
3           # scout limit
10.00000000 # amplitude
ef -0       # save optimized configuration
30          # number of LMs to be saved
```

```
2
charmm36/hno3.xyz  1
charmm36/tip4p.xyz 10
* 10.0000
```

From ABCluster 2.0, you can add external electric field to the cluster. Now we want to see how the cluster will deform under the influence of electric field. This is very easy! Assume the electric field strength is 0.8 V Å\(^{-1}\), we prepare the input files called `ef-8.inp` and `ef-8.cluster`:

Warning: The electric field can ONLY be applied when no charged species exist!

```
ef-8.cluster # cluster file name
30          # population size
50          # maximal generations
3           # scout limit
10.00000000 # amplitude
ef -8       # save optimized configuration
30          # number of LMs to be saved
```

```
2 0.8
charmm36/hno3.xyz  1
charmm36/tip4p.xyz 10
* 10.0000
```

Note that, the electric field strength \(F\) is added in the first line of `ef-8.cluster`. If it is a single integer like 2, it will indicate that there is no external electric field. If it is an integral and a real number like 2 0.8, it means that \(F = 0.8\) V Å\(^{-1}\).

Now run the global optimization:

```
rigidmol-optimizer ef-8.inp > ef-8.out
```

We can see that the global minimum is stretched along the \(z\) direction.
4.7. EXAMPLE 6: \(\text{HNO}_3(\text{H}_2\text{O})_{10}\) IN AN ELECTRIC FIELD

Figure 4.7.1: The GM of \(\text{HNO}_3(\text{H}_2\text{O})_{10}\) in an electric field.
Chapter 5

Optimization with Third-party Programs

5.1 Introduction

In the last two chapters, we discussed several kinds of force fields which are supported natively by ABCluster. However, for species which exhibits strong many-body effects, like inorganic clusters or small metal clusters, to the best of the author’s knowledge, no force fields can describe them reasonably. In these cases, one must use quantum chemistry instead of additive empirical force field potentials to calculate their energies. The component isomer and lego of ABCluster are designed for this purpose. The program itself provides some simple interfaces. By using these interfaces one can perform the global optimization with energy calculated by any third-party programs, e.g. Gaussian, xTB-GFN, DMol3, VASP. Since these calculations are much more expensive than those of force fields, the ABC algorithm is simplified. From ABCluster 2.0, the generation of initial guess has been significantly improved, and the program has been parallelized for multi-nodes. We will now show how to combine the power of ABCluster working with other programs.

5.2 ABCluster with Gaussian

Gaussian is one of the most popular quantum chemistry software, thus we have provided official support for it. In the distribution of ABCluster, you can find two files: xyz2gaussian and gaussian2xyz. They will be used by ABCluster to compute the energy with Gaussian.

5.2.1 Configuration of Gaussian

One must do some preparations before using Gaussian, e.g., setting up some environmental variables. If you have done this you can skip this section.

For Windows users: Assume you use Gaussian09 and put it in the directory D:\G09W, then open Start→Control Panel→System or Security→System, then click Advanced System Settings, choose the Environment Variables option. In the User variables for you list, click New. In the Variable name editbox, write “GAUSS_EXEDIR”, and in the Variable value editbox, write “D:\G09W”. Then click OK to confirm it.

For Linux and Mac OS X users: Assume you use Gaussian09 and put it in the directory /home/you/g09 and you set the scratch directory for the computation as /scratch/you. then open your .bashrc. At the end of the file, add the statements:
CHAPTER 5. OPTIMIZATION WITH THIRD-PARTY PROGRAMS

export g09root="/home/you/g09"
export GAUSS_SCRDIR=/scratch/you
source $g09root/g09/bsd/g09.profile

5.2.2 Example 1: Al$_3$O$_4^+$

The first example is, what is the global minimum of the cluster Al$_3$O$_4^+$? We decide to use a classical method: B3LYP/6-31g(d) to calculate energy. Now we will do this global optimization.

Step 1: In the ABCluster distribution, go to the directory testfiles/qc. Create a file called mol.inp. For Windows users, the content should be:

```
mol            # Result file name
Al 3 O 4       # Symbols
cube 3 3 3     # Structure types
30             # Maximal number of calculations

>>>>
"D:\G09W\g09.exe" $xxx$.gjf
../gaussian2xyz $xxx$.out > $out$
```

For Linux and Mac OS X users,

```
mol            # Result file name
Al 3 O 4       # Symbols
cube 3 3 3     # Structure types
30             # Maximal number of calculations

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

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 cluster components. For Al$_3$O$_4^+$, just write Al 3 O 4. For Au$_{13}$Pt$_1$, write Au 13 Pt 1.
- **Line 3**: For this line, one can give the type of initial guesses. Available choices are listed below and are shown in Figure 5.2.1:
  - line Generate a linear initial structure.
  - ring Generate a ring initial structure.
  - sphere Generate a sphere initial structure.
  - plane 2 3 Generate a 2D initial structure from a 2 × 3 lattice.
  - cube 3 4 3 Generate a 3D initial structure from a 3 × 4 × 3 lattice.

Note that this does not mean that the program will only give linear structures when you use line. It is just a “preference”.
5.2. ABCLUSTER WITH GAUSSIAN

- Line 4: The maximal number of calculations you want to do. Here it means `isomer` will do 30 calculations.

- The lines between two `>>>>` are the commands to call third-party programs to calculate the energy. This will be explained latter.

Figure 5.2.1: The types of initial guesses.

Step 2: Create a file named `optfile` with the following content:

```
% nproc = 20
% mem = 300 GB
#N B3LYP/6-31g(d) scf(xqc, novaracc) opt(MaxCycles=100)

b opt
+1 1
>>>>

>>>>
```

Note that you should modify `nproc` and `mem` for your machine.

Step 3: Run the global optimization:

```
nohup ../../isomer mol.inp > mol.out &
```

Note that this optimization is very expensive. During the optimization, there will be a directory named `mol-LM`, in which one can check the found local minima so far. Also, one can check the output file `mol.out`, where one can see the current energies. An example output is:

```
Will perform at most 30 calculations.
-------------------------------------------
```
CHAPTER 5. OPTIMIZATION WITH THIRD-PARTY PROGRAMS

### Energy Time State

<table>
<thead>
<tr>
<th>#</th>
<th>Energy</th>
<th>Time</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-1028.04693646</td>
<td>397</td>
<td>Succeed</td>
</tr>
<tr>
<td>1</td>
<td>-899.91003266</td>
<td>51</td>
<td>Succeed</td>
</tr>
<tr>
<td>2</td>
<td>-1028.05070398</td>
<td>380</td>
<td>Succeed</td>
</tr>
<tr>
<td>3</td>
<td>-1027.95122573</td>
<td>173</td>
<td>Succeed</td>
</tr>
<tr>
<td>4</td>
<td>-1027.98390199</td>
<td>377</td>
<td>Succeed</td>
</tr>
<tr>
<td>5</td>
<td>-1028.08693656</td>
<td>219</td>
<td>Succeed</td>
</tr>
<tr>
<td>6</td>
<td>-1028.08350594</td>
<td>414</td>
<td>Succeed</td>
</tr>
<tr>
<td>7</td>
<td>-1027.94655101</td>
<td>357</td>
<td>Succeed</td>
</tr>
<tr>
<td>8</td>
<td>-1027.84124312</td>
<td>260</td>
<td>Succeed</td>
</tr>
</tbody>
</table>

This means, the program is performing the 10th calculations. You can check the structure with low energy $-1028.08693656$ by the file mol-LM/5.xyz. See Figure 5.2.2.

Step 4: After the optimization, you can check all the structures. The 27th structure has the lowest energy $-1028.18359791$ and is the global minimum. You can check it from 27.xyz. The structures are shown in Figure 5.2.2.

![Figure 5.2.2: One local and global minimum of Al$_3$O$_4^+$](image)

-1028.08693656  -1028.18359791

5.2.3 More About optfile

Now we describe what optfile is. It is used by xyz2gaussian to convert a XYZ molecule into a Gaussian input file. Here is an example optfile. xyz2gaussian will put all the lines before the first >>> in front of the coordinates, and all the lines between the two >>> at the end of the coordinates, to generate a Gaussian input file.

In practice, you have to modify the charge, spin multiplicity, method, basis set, and other options to satisfy your requirement. Here scf(xqc, novaracc) is recommended since the initial guess can be very distorted, and this option can guarantee the convergence of the SCF calculations. For their meanings, please refer to the Gaussian manual.

5.2.4 Example 2: Au$_6$

Now we want to know the global minimum of Au$_6$. Maybe you are thinking about the Gupta or Sutton–Chen potential. Unfortunately, for small metallic clusters, both potentials are very inaccurate[23]. Thus quantum chemistry must be used. Also, gold “enjoys” very strong relativistic effects. Thus one should use a suitable pseudopotential to describe gold. A good choice is the ECP60MDF. This pseudopotential can be obtained from [http://www.tc.uni-koeln.de/PP/clickpse.en.html](http://www.tc.uni-koeln.de/PP/clickpse.en.html).
Step 1: In the ABCluster distribution, go to the directory testfiles/qc. Create a file called mol.inp. For Windows users, the content should be:

```
mol      # Result file name
Au 6     # Symbols
plane 4 3 # Structure types
30       # Maximal number of calculations
>>>>
../../xyz2gaussian optfile $inp$ > $xxx$.gjf
"D:\G09\g09.exe" $xxx$.gjf
../../gaussian2xyz $xxx$.out > $out$
>>>>
```

For Linux and Mac OS X users,

```
mol      # Result file name
Au 6     # Symbols
plane 4 3 # Structure types
30       # Maximal number of calculations
>>>>
../../xyz2gaussian optfile $inp$ > $xxx$.gjf
g09 < $xxx$.gjf > $xxx$.log 2>/dev/null
../../gaussian2xyz $xxx$.log > $out$
>>>>
```

Since I know that the global minimum of small gold clusters should be plane, I use plane 4 3 for you to save time. In practice, however, when you do not know what the global minimum should be, you’d better use cube to search more kinds of structures.

Step 2: Create a file named optfile with the following content:

```
%nproc=20
%mem=400GB
#N TPSSTPSS/GENECP scf(xqc,novaracc) opt(Cartesian, MaxCycles=100)

b opt

0 1
>>>>
Au 0
S 8 1.00
0.38000800E+02 0.20009000E-01
0.23972500E+02 -0.15167900E+00
0.15218200E+02 0.36396000E+00
0.55399900E+01 -0.82132600E+00
0.15649600E+00 0.93664100E+00
0.64246100E+00 0.42352700E+00
0.15649600E+00 0.16250000E-01
0.54910000E-01 -0.97800000E-03
S 8 1.00
0.38000800E+02 -0.53040000E-02
0.23972500E+02 0.46318000E-01
0.15218200E+02 -0.11994000E+00
0.55399900E+01 0.30406200E+00
```
| 0.13855100E+01 | -0.49450100E+00 |
| 0.64246100E+00 | -0.25551600E+00 |
| 0.15649600E+00 | 0.59776700E+00 |

S 8 1.00

| 0.38000800E+02 | 0.32340000E-01 |
| 0.23972500E+02 | -0.17176200E+00 |
| 0.15218200E+02 | 0.33950300E+00 |

| 0.13855100E+01 | 0.19174980E+01 |
| 0.64246100E+00 | -0.11299310E+01 |
| 0.15649600E+00 | -0.13662840E+01 |

| 0.54910000E-01 | 0.14423910E+01 |

S 1 1.00

| 0.54910000E-01 | 0.10000000E+01 |

P 7 1.00

| 0.10309200E+02 | 0.12821700E+00 |
| 0.66276500E+01 | -0.35379000E+00 |
| 0.16744700E+01 | 0.56621100E+00 |

| 0.80111600E+00 | 0.49317100E+00 |
| 0.34687900E+00 | 0.11377500E+00 |
| 0.12270100E+00 | 0.25340000E-02 |

| 0.42428000E-01 | 0.67900000E-03 |

P 7 1.00

| 0.10309200E+02 | -0.35885000E-01 |
| 0.66276500E+01 | 0.10289000E+00 |
| 0.16744700E+01 | -0.20098800E+00 |

| 0.80111600E+00 | -0.20104500E+00 |
| 0.34687900E+00 | 0.11654900E+00 |
| 0.12270100E+00 | 0.59496900E+00 |
| 0.42428000E-01 | 0.45552000E+00 |

P 7 1.00

| 0.10309200E+02 | -0.77242000E-01 |
| 0.66276500E+01 | 0.22261100E+00 |
| 0.16744700E+01 | -0.49535300E+00 |

| 0.80111600E+00 | -0.36074700E+00 |
| 0.34687900E+00 | 0.71000000E+00 |
| 0.12270100E+00 | 0.56883900E+00 |
| 0.42428000E-01 | 0.11930000E-02 |

P 1 1.00

| 0.42428000E-01 | 0.10000000E+01 |

D 6 1.00

| 0.11002700E+02 | 0.16467000E-01 |
| 0.68916600E+01 | -0.68013000E-01 |
| 0.18080800E+01 | 0.29949200E+00 |

| 0.82105100E+00 | 0.45430300E+00 |
| 0.34416100E+00 | 0.34422400E+00 |
| 0.12974300E+00 | 0.12125600E+00 |

D 6 1.00

| 0.11002700E+02 | -0.23628000E-01 |
| 0.68916600E+01 | 0.95672000E-01 |
Note that here we use TPSS/ECP60MDF method.

Step 3: Run the global optimization:

```
nohup ../../isomer mol.inp > mol.out &
```

Step 4: After the optimization, you can check the structures. For my run, the first structure (mol-LM/0.xyz) happens to be the global minimum, which is shown in Figure 5.2.3.
5.3 ABCluster with Other Programs

ABCluster can in principle work with any programs. Now we will see how to do this.

5.3.1 How does isomer and lego work?

When isomer is running, it will generate a geometry with a long name, like x-34. You do not need to know its exact name, in ABCluster it can be represented by $xxx$. This geometry will be saved in standard XYZ format too a file named x-34.xxxxxyz, represented by $inp$. Then, commands between the two >>> will be run one by one. Finally, the optimized geometry and energy should be written in XYZ format to a file named x-34.ooooout (The title line is the energy!), represented by $out$. Then isomer will read optimized geometry and energy from this file and store it. Then isomer will start next search. This mechanism can be shown in Figure 5.3.1.

Figure 5.3.1: How isomer work with third-party programs.
5.3. ABCLUSTER WITH OTHER PROGRAMS

Now let’s see how we let ABCluster work with Gaussian. For the following commands:

```plaintext
>>>>
../../xyz2gaussian optfile $inp$ > $xxx$.gjf
g09 < $xxx$.gjf > $xxx$.log 2>/dev/null
../../gaussian2xyz $xxx$.log > $out$
>>>>
```

In the first line, `xyz2gaussian` use the structure (`$inp$`) generated by `isomer` and `optfile` to generate a Gaussian input file named `$xxx$.gjf`. The second line calls Gaussian to perform the local optimization, the output of which is in `$xxx$.log`. The third line extracts the final optimized structure and corresponding energy and saves it in `$out$`. This is exactly what Figure 5.3.1 instructs you to do.

In summary, your main task is to write a program or script to translate the XYZ format geometry into the format that the third-party program accepts, and perform the calculation, then extract energy and optimized geometry and save them in another XYZ file.

5.3.2 Example 3: ABCluster with ORCA on He$_2$Ne$_2$

Assuming you work in Linux and want to use ORCA to perform the quantum chemistry calculation. You want to know the global minimum of a very weakly bound cluster He$_2$Ne$_2$.

Step 1: In the ABCluster distribution, go to the directory `testfiles/qc`.

Step 2: In Linux, one can write a shell script to generate the input file, do the calculation, and extract the energy and geometry information. One possible script is:

```bash
#!/bin/bash

# Create input file
fn=$1-orca
xyzfn=$2
outfn=$3
echo "" ! B3LYP D3BJ TZVPP TZVPP/JK RIJCOSX TightSCF OPT
  * xyz 0 1":" $fn.inp
awk 'NR>2{ print }' $xyzfn >> $fn.inp
echo "" >> $fn.inp

# Run the calculation
/nfs/software/chemie/orca/3.0.3/linux_x86-64/orca $fn.inp
  > $fn.out

# Transform the output file
energy=`grep "Total Energy" $fn.out | tail -n 1 | awk '{
  print $4}``

ded "2s/".*$/energy/" $fn.xyz > $outfn
rm -rf $fn.engrad $fn.gbw $fn.opt $fn.prop $fn.trj
  $fn.xyz
```

If you are familiar with shell programming and ORCA input/output file format, this file is self-explaining. We call it as `oa`. Once create it, you must give an “executable” permission.
chmod +x oa

Step 2: Prepare a file called mol.inp:

```
mol # Result file name
He 2 Ne 2 # Symbols
cube 2 2 2 # Structure types
10 # Maximal number of calculations
>>>>
./oa $xxx$ $inp$ $out$
>>>>
```

Step 3: Run the optimization:

```
nohup ../../isomer mol.inp > mol.out &
```

Step 4: Check the output file mol.out, it seems that structure 3 is the most stable one, shown in Figure 5.3.2.

```
<table>
<thead>
<tr>
<th>#</th>
<th>Energy</th>
<th>Time</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-263.68047485</td>
<td>233</td>
<td>Succeed</td>
</tr>
<tr>
<td>1</td>
<td>-263.68100419</td>
<td>277</td>
<td>Succeed</td>
</tr>
<tr>
<td>2</td>
<td>-263.68070979</td>
<td>287</td>
<td>Succeed</td>
</tr>
<tr>
<td>3</td>
<td>-263.68123254</td>
<td>242</td>
<td>Succeed</td>
</tr>
<tr>
<td>4</td>
<td>-263.68029763</td>
<td>307</td>
<td>Succeed</td>
</tr>
<tr>
<td>5</td>
<td>-263.68116986</td>
<td>274</td>
<td>Succeed</td>
</tr>
<tr>
<td>6</td>
<td>-263.68101136</td>
<td>293</td>
<td>Succeed</td>
</tr>
<tr>
<td>7</td>
<td>-263.68053291</td>
<td>248</td>
<td>Succeed</td>
</tr>
<tr>
<td>8</td>
<td>-263.68026220</td>
<td>119</td>
<td>Succeed</td>
</tr>
<tr>
<td>9</td>
<td>-263.68019605</td>
<td>260</td>
<td>Succeed</td>
</tr>
</tbody>
</table>
```

5.3.3 Support For ORCA, NWChem, xTB-GFN, and DMol3

We provided official support for ABCluster with ORCA, NWChem, xTB-GFN, and DMol3. Due to the limiation of these programs, these supports are only available for the Linux version of ABCluster.

For ORCA and NWChem, we provide two scripts, `orca4abcluster.sh` and `nwchem4abcluster.sh`. You can easily modify them to adjust to your own system and requirement. For usage, you can refer to Subsection 5.3.2, where you can simply replace `oa` to `orca4abcluster.sh` or `nwchem4abcluster.sh`. Make sure that these two scripts as well as the commands in them are available in `$PATH`.

For DMol3, we provide one script, `dmol34abcluster.sh` (Courtesy of Mr. Kai Wang).

For xTB-GFN, you can refer to Subsection 5.4.2.

5.4 Examples for lego

From ABCluster 1.4, a new component `lego` was added. It can do the optimization with not only atoms but also molecular fragments.
5.4.1 Example 4: (NH$_3$)$_4^-$

We here give an example to show how to use lego with Gaussian. You will find it very similar to isomer.

For (NH$_3$)$_4^-$, one can use rigidmol-optimizer with CHARMM36 force field to do the search. However, when an extra electron is solvated, it is not trivial to use force field any more. Instead, we have to use quantum chemistry to calculate its energy. Here, we will combine ABCluster with Gaussian09, using B3LYP/SVP method.

Step 1: In the ABCluster distribution, go to the directory testfiles/qc. Create a file called enh34.inp. For Linux and Mac OS X users, it looks like:

```plaintext
enh34 # Result file name
enh34 . cluster # Cluster file name
cube 2.5 2 2 2 # Structure types
20 # Maximal number of calculations

>>>>
../ xyz2gaussian optfile $inp$ > $xxx$.gjf
g09 < $xxx$.gjf > $xxx$.log 2>/dev/null
../ gaussian2xyz $xxx$.log > $out$

```

The meaning of each line:

- **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 cluster component file name.
- **Line 3**: For this line, one can give the type of initial guesses. Available choices are:
  - **line 3.5** Generate a linear initial structure with molecule spacing 3.5 Å.
  - **ring 3.5** Generate a ring initial structure with a radius 3.5 Å.
– sphere 3.5 Generate a sphere initial structure with radius 3.5 Å. The first molecule will be solvated and fixed at the center.

– plane 2.5 2 3 Generate a 2D initial structure from a $2 \times 3$ lattice, with the distance between neighboring points being 2.5 Å.

– cube 2.5 3 4 3 Generate a 3D initial structure from a $3 \times 4 \times 3$ lattice, with the distance between neighboring points being 2.5 Å.

– surface +10.5 3.5 Generate an “absorption” structure. See Section 5.5.

Note that this does not mean that the program will only give linear structures when you use line. It is just a “preference”.

- Line 4: The maximal number of calculations you want to do. Here it means lego will do 20 calculations.

- The lines between two >>> are the commands to call third-party programs to calculate the energy. This will be explained latter.

Step 2: Create a file named optfile with the following content:

```bash
% nproc = 8
% mem = 30 GB
#N B3LYP/6-31++g(d) scf(xqc, novaracc) opt(MaxCycles = 100)
e(NH3)4
-1 2
>>>>
>>>>
```

Note that you should modify nproc and mem for your machine.

Step 3: Create a file named enh34.cluster with the following content. Its format is identical to that for rigidmol-optimizer.

```bash
1
./nh3.xyz 4
```

Step 4: Create a file named nh3.xyz to give its structure in standard XYZ format.

```plaintext
4
NH3
N  -0.0000000  -0.0000000  -0.10000001
H   0.0000000   0.9428090  0.23333324
H  -0.81649655  0.47140450  0.23333324
H   0.81649655  0.47140450  0.23333324
```

Step 5: Run the global optimization:

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

When the optimization is accomplished, there will be a directory named enh34-LM, in which one can check the found local minima so far. Also, one can check the output file enh34.out, where one can see the current energies. An example output is:
5.4. EXAMPLES FOR LEGO

 WILL PERFORM AT MOST 20 CALCULATIONS.

<table>
<thead>
<tr>
<th>#</th>
<th>Energy</th>
<th>Time</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.00000000</td>
<td>6</td>
<td>The structure is not</td>
</tr>
<tr>
<td>1</td>
<td>-225.99332646</td>
<td>104</td>
<td>Succeed</td>
</tr>
<tr>
<td>2</td>
<td>-226.00850994</td>
<td>636</td>
<td>Succeed</td>
</tr>
<tr>
<td>3</td>
<td>-226.00185328</td>
<td>292</td>
<td>Succeed</td>
</tr>
<tr>
<td>4</td>
<td>-226.00865554</td>
<td>616</td>
<td>Succeed</td>
</tr>
<tr>
<td>5</td>
<td>-226.00852078</td>
<td>651</td>
<td>Succeed</td>
</tr>
<tr>
<td>6</td>
<td>-225.98549663</td>
<td>242</td>
<td>Succeed</td>
</tr>
<tr>
<td>7</td>
<td>-226.00842890</td>
<td>337</td>
<td>Succeed</td>
</tr>
<tr>
<td>8</td>
<td>-225.98622261</td>
<td>124</td>
<td>Succeed</td>
</tr>
<tr>
<td>9</td>
<td>-226.00425905</td>
<td>617</td>
<td>Succeed</td>
</tr>
<tr>
<td>10</td>
<td>-226.00495131</td>
<td>190</td>
<td>Succeed</td>
</tr>
<tr>
<td>11</td>
<td>-225.99694564</td>
<td>101</td>
<td>Succeed</td>
</tr>
<tr>
<td>12</td>
<td>-225.99427656</td>
<td>542</td>
<td>Succeed</td>
</tr>
<tr>
<td>13</td>
<td>-226.00424133</td>
<td>592</td>
<td>Succeed</td>
</tr>
<tr>
<td>14</td>
<td>-226.00833984</td>
<td>330</td>
<td>Succeed</td>
</tr>
<tr>
<td>15</td>
<td>-226.00862608</td>
<td>639</td>
<td>Succeed</td>
</tr>
<tr>
<td>16</td>
<td>-226.00493762</td>
<td>151</td>
<td>Succeed</td>
</tr>
<tr>
<td>17</td>
<td>-225.98721389</td>
<td>104</td>
<td>Succeed</td>
</tr>
<tr>
<td>18</td>
<td>-225.98747996</td>
<td>81</td>
<td>Succeed</td>
</tr>
<tr>
<td>19</td>
<td>-226.00438437</td>
<td>272</td>
<td>Succeed</td>
</tr>
</tbody>
</table>

All the calculations are finished!

The No. 15 structure (Energy: -226.00862608) seems to be the most stable one. However, as we emphasize for several times, you must always study the optimized structures with more advanced quantum chemistry. (NH₃)₄ can be a very diffuse species, thus an acceptable study must include very diffuse basis functions. Also, for such systems, density functional theory is not reliable, MP2 is a better choice. We will not do this job here. Instead, we show its global minimum as well as the singly occupied molecular orbital (SOMO) in Figure 5.4.1.

![Figure 5.4.1](image_url)  

**Figure 5.4.1:** The global minimum of (NH₃)₄⁻, and its SOMO (isovalue: 0.02 A.U.).
5.4.2 Example 5: (CO)(H\(_2\)O)\(_5\) Using xTB-GFN

xB-GFN is a fast and accurate method developed by Prof. Stefan Grimme\[24\]. It can work with ABCluster perfectly. You can find examples in testfiles/qc/5-xtb. Here we want to show how to use xTB-GFN and lego to optimize the cluster (CO)(H\(_2\)O)\(_5\).

Like the example above, you should prepare the geometry of CO and H\(_2\)O, in co.xyz:

```
2
CO
C 0 0 0
0 0 0 1.4
```

and h2o.xyz:

```
3
water
0 0.00000000 0.00000000 0.21451132
H 0.00000000 0.75695033 -0.37137096
H 0.00000000 -0.75695033 -0.37137096
```

and the cluster file x.cluster:

```
2
c0.xyz 1
h2o.xyz 5
```

Now, prepare the input file x.inp:

```
x # Result file name
x.cluster # Cluster file name
cube 2.5 4 3 3 # Structure types
200 # Maximal number of calculations

>>>>
cp $inp$ $xxx$.xyz
txb $xxx$.xyz -opt > $xxx$.out
cp xtbopt.coord $out$

```}

For the usage of xtb, please refer to its manual. Now run the global optimization by:

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

After several minutes, you can see this at the end of x.out:

```
# Energy

91  -35.64398705
133 -35.64390478
68  -35.64382130
82  -35.64365158
6   -35.64293376
104 -35.64185713
134 -35.64185642
```
5.5. Example 6: Absorption

From ABCluster 1.5, lego is able to generate structures that are suitable for studying surface absorption. We will only give a brief description here. Assuming you want to study the gas absorption on metal surface, say H₂ and CO on Pt surface. The cluster file mol.cluster should be written as:

```
3
pt.xyz 1
h2.xyz 3
c0.xyz 3
```

where the surface must come before any absorbed molecules. pt.xyz contains the coordinates of the surface, which could be a slab of metal like Figure 5.5.1. In order to give a well-defined surface, the surface at which the molecules are supposed to absorb should be vertical to z-axis with the coordinate origin at (0,0,0). In the input file:

```
mol # Result file name
mol.cluster # Cluster file name
surface +10.5 3.5 # Structure types
20 # Maximal number of calculations

...>
```

where surface +10.5 3.8 gives the details of the surface configuration: the molecules should be absorbed in the range of z > 10.5 and -3.5 < x,y < +3.5, as illustrated in Figure 5.5.1.

You can further optimize several low energy structures using sophisticated quantum chemical methods. The structure of x=LM/91.xyz is shown in Figure 5.4.2.
The generated structures can be further optimized by Gaussian, VASP, LAMMPS, etc.

5.6 Parallelization

From ABCluster 2.0, isomer and lego can use multi-nodes to accelerate the global optimization. There is a mini queueing system in isomer and lego that can distribute calculations on all nodes. Below we will show how to set up this parallelization.

Before doing the parallelization on different nodes, you should make sure that:

- The file systems on all nodes are identical (Say, using NFS or Lustre).
- All nodes can be connected using SSH without password.

If you do not know how to build a cluster like this, please contact your computer administrator or check the author’s website:

http://www.zhjun-sci.com/theochem-clusterbuilding-EN.php

5.6.1 Example 7: Zn$_3$S$_3$ with Parallelization

Assume you want to know the structure of Zn$_3$S$_3$. We can use isomer and provide the following input files:

**optfile**: We will optimize Zn$_3$S$_3$ using B3LYP/6-31G(d) with Gaussian09.

```
%mem=20GB
%B3LYP/6-31g(d) scf(xqc, novaracc) opt(MaxCycles=100)
```

```
0 1
```
5.6. PARALLELIZATION

zns.inp:

```plaintext
zns      # Result file name
Zn 3 S 3  # Cluster file name
cube 3 3 3 # Structure types
30       # Maximal number of calculations

xyz2gaussian 'pwd'/optfile $inp$ > $xxx$.gjf
g09 < $xxx$.gjf > $xxx$.log 2>/dev/null
gaussian2xyz $xxx$.log > $out$
rm $xxx$.gjf $xxx$.log

nodes
```

Note that, in the last line of the input file, **if there is nothing behind the last >>>, the program will run automatically on a single node as before. Otherwise, if there is file name, like nodes here (it can be an any name.), the program will run in a parallel mode.**

Also, note that, in the case of parallelization, the files involved in commands between two >>>’s must contain **absolute path** (for example, see xyz2gaussian ‘pwd’/optfile). But ABCluster internal variables like $xxx$ will add absolute path automatically, so they do not bother you.

Here is the file nodes. We assume that you have three computers in your cluster, each having 16 CPU cores and 20GB memory.

3
node001
node002
node003

This file means: ABCluster will use the three computers of hostname node001, node002 and node003 to do the calculations. According to the optfile, on each node, Gaussian09 will use 15 CPU cores and 20 GB memory. If you think that, on each node, it is more efficient for Gaussian09 to use 8 CPUs and 10GB, you can use the following configuration:

6
node001
node001
node002
node002
node003
node003

now, run isomer as usual:

nohup isomer zns.inp > zns.out &

By checking out zns.out, we can see that the program is indeed parallelized:

```plaintext
-- Parallelization --
Parallelization with:
```
--- Isomer Optimization ---
Will perform at most 30 calculations.

<table>
<thead>
<tr>
<th>#</th>
<th>Energy</th>
<th>Time</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-6532.01163106</td>
<td>350</td>
<td>Succeed</td>
</tr>
<tr>
<td>2</td>
<td>-6531.95110987</td>
<td>451</td>
<td>Succeed</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Reordered from low to high energy:

<table>
<thead>
<tr>
<th>#</th>
<th>Energy</th>
</tr>
</thead>
<tbody>
<tr>
<td>22</td>
<td>-6532.11786138</td>
</tr>
<tr>
<td>20</td>
<td>-6532.03179201</td>
</tr>
<tr>
<td>18</td>
<td>-6532.01920253</td>
</tr>
<tr>
<td>11</td>
<td>-6531.95941532</td>
</tr>
<tr>
<td>24</td>
<td>-6531.95110987</td>
</tr>
<tr>
<td>5</td>
<td>-6531.93104687</td>
</tr>
<tr>
<td>29</td>
<td>-6531.93038886</td>
</tr>
<tr>
<td>21</td>
<td>-6531.92802462</td>
</tr>
<tr>
<td>26</td>
<td>-6531.92756921</td>
</tr>
<tr>
<td>8</td>
<td>-6531.90616210</td>
</tr>
<tr>
<td>27</td>
<td>-6531.89530297</td>
</tr>
</tbody>
</table>

Figure 5.6.1 is the final global minimum 22.xyz:

5.6.2 Example 8: Fe(H₂O₂)_3²⁺ with Parallelization

lego can be parallelized in the same way as isomer. For example, Fe(H₂O₂)_3²⁺. We prepare the node file nodes:

3
node001
node002
node003

and mol.inp:

mol # Result file name
mol.cluster # Cluster file name
sphere 3.5 # Structure types
30 # Maximal number of calculations

>>>
5.6. PARALLELIZATION

Figure 5.6.1: The global minimum of Zn$_3$S$_3$.

```
xyz2gaussian 'pwd'/'optfile $inp$ > $xxx$.gjf
g09 < $xxx$.gjf > $xxx$.log 2>/dev/null
gaussian2xyz $xxx$.log > $out$
rm $xxx$.gjf $xxx$.log

>>>>

This is the cluster file mol.cluster:

```
2
./fe.xyz 1
./h2o2.xyz 3
```

and fe.xyz and h2o2.xyz:

```
1
Fe
Fe 0. 0. 0.

4
H2O2
0  -0.65334587 -0.09348352 -0.07454123
H  -1.05617580 0.46262807 0.59632982
O  0.65334587  0.09348352 -0.07454123
H  1.05617580 -0.46262807  0.59632982
```

Finally, optfile. Here, we simply assume that the spin of the Fe$^{2+}$ is $S = 2$.

```
%nproc=15
```
% mem = 20 GB
#N B3LYP/6-31g(d) scf(xqc,novaracc) opt(MaxCycles=100)

opt
+2 5
>>>>

Now run lego:

nohup lego mol.inp > mol.out &

This job will be parallelized on the three nodes.

Figure 5.6.2 is the final global minimum .xyz:

![Figure 5.6.2: The global minimum of Fe(H₂O₂)²⁺.](image-url)
Chapter 6

Coarse-Grained Particle Clusters

6.1 Introduction

From ABCluster 1.5, a new component cg-optimizer is added to ABCluster, which enables the optimization of clusters consisting of anisotropic particles. The particle here is not a simple point treated in previous chapters, but can be viewed as a coarse-grained model of a small or large chemical units. For example, a benzene, instead of treating as a 12-atom molecule, is modeled by a particle with a nonzero quadrupole moment. To make this possible, each particle has 6 rather than 3 degrees of freedom: position \( \mathbf{r} = \{x, y, z\} \) and its orient vector \( \mathbf{n} = \{n_x, n_y, n_z\} \). Note that the orient vector is always a unit one, satisfying \( |\mathbf{n}| = 1 \), due to this constraint the number of degrees of freedom is strictly 5.

The visualization of these coarse-grained particles can be done by a small software written by the author, cgviewer. Please refer to Section 6.6.

In ABCluster, there are several kinds of coarse-grained particles available to build models of large chemical systems:

1. Lennard-Jones–Coulomb (LJC).
2. Dipole moment (DM).
3. Quadrupole moment (QM).

The interactions between these particles are listed below:

1. LJC-LJC: The interaction is exactly the same as (4.1.1) in Chapter 4, and it does not depend on \( \mathbf{n}_i \) and \( \mathbf{n}_j \):

\[
U_{\text{LJC-LJC}} = 4\epsilon_{ij} \left( \left( \frac{\sigma_{ij}}{r_{ij}} \right)^{12} - \left( \frac{\sigma_{ij}}{r_{ij}} \right)^{6} \right) + \frac{e^2}{4\pi\epsilon_0} \frac{q_i q_j}{r_{ij}} \quad (6.1.1)
\]

2. LJC-DM: This is the interaction between a point charge \( i \) and a dipole moment \( j \):

\[
U_{\text{CD}} = -\frac{e^2}{4\pi\epsilon_0} q_i d_j \frac{\mathbf{n}_j \cdot \mathbf{r}_{ij}}{r_{ij}^3} \quad (6.1.2)
\]

where

\[
\mathbf{r}_{ij} = \mathbf{r}_j - \mathbf{r}_i \quad (6.1.3)
\]
3. DM-DM: The interaction between two dipole moments:

\[
U_{DD} = \frac{e^2}{4\pi\epsilon_0} d_i d_j \left( \frac{n_i \cdot n_j}{r_{ij}^3} - \frac{3(n_i \cdot r_{ij})(n_j \cdot r_{ij})}{r_{ij}^5} \right) \tag{6.1.4}
\]

4. LJC-QM: The interaction between a charge and quadrupole moment:

\[
U_{CQ} = \frac{e^2}{4\pi\epsilon_0} q_i Q_j \left( \frac{3(n_j \cdot r_{ij})^2}{r_{ij}^5} - \frac{1}{r_{ij}^3} \right) \tag{6.1.5}
\]

5. DM-QM: The interaction between a dipole and quadrupole moment:

\[
U_{DQ} = \frac{e^2}{4\pi\epsilon_0} 3d_i Q_j \left( \frac{1 + 2(n_i \cdot n_j)^2}{r_{ij}^5} - 5(n_i \cdot r_{ij})^2 + (n_j \cdot r_{ij})^2 + 4(n_i \cdot r_{ij})(n_j \cdot r_{ij})(n_i \cdot n_j) \right) \tag{6.1.6}
\]

6. QM-QM: The interaction between two quadrupole moments:

\[
U_{QQ} = \frac{e^2}{4\pi\epsilon_0} 3Q_i Q_j \left( \frac{1 + 2(n_i \cdot n_j)^2}{r_{ij}^5} - \frac{3(n_i \cdot r_{ij})(n_j \cdot r_{ij})^2}{r_{ij}^5} + 35 \right) \tag{6.1.7}
\]

7. PY-PY: This is a kind of anisotropic Lennard-Jones potential. Before describing Paramonov–Yaliraki potential we first introduce elliptic contact function (ECF)[25]. Consider two ellipsoids A and B with the lengths of their center and semiaxes \( r_A, a_1, a_2, a_3 \) and \( r_B, b_1, b_2, b_3 \), respectively. Their orientation are given by three orthonormal unit vectors: \( u_1, u_2, u_3 \) and \( v_1, v_2, v_3 \), respectively. Now we view \( r, u \) and \( v \) as column matrices, then the ellipsoid equations are:

\[
S_A(r) = (r - r_A)^T A (r - r_A) \tag{6.1.8}
\]

\[
S_B(r) = (r - r_B)^T B (r - r_A) \tag{6.1.9}
\]

where

\[
A \equiv a_1^{-2} u_1 u_1^T + a_2^{-2} u_2 u_2^T + a_3^{-2} u_3 u_3^T \tag{6.1.10}
\]

\[
B \equiv b_1^{-2} v_1 v_1^T + b_2^{-2} v_2 v_2^T + b_3^{-2} v_3 v_3^T \tag{6.1.11}
\]

Now we define such an auxilliary function:

\[
S(r, \lambda) = \lambda S_A(r) + (1 - \lambda) S_B(r), \quad \lambda \in [0, 1] \tag{6.1.12}
\]

Then we define the ECF:

\[
F(A, B) = \max_{\lambda} \min_r S(r, \lambda) \tag{6.1.13}
\]

If \( F(A, B) < 1 \), the two ellipsoids overlap, if \( F(A, B) > 1 \), they do not. We also give the analytical derivatives of the ECF, since their derivations are not so trivial. Assume in (6.1.13), the final optimized \( r \) and \( \lambda \) are denoted by \( r_C \) and \( \lambda_C \), then

\[
\frac{\partial}{\partial r_A} F(A, B) = -2\lambda_C A (r_C - r_A) \tag{6.1.14}
\]
6.2. Example 1: Why (HF)$_2$ is nonlinear?

HF is obviously a polar molecule, having a dipole moment of 0.72 au. It may be thought that the HF dimer, (HF)$_2$, should be linear. However, it is not, see Figure 6.2.1. Why? Of course, we can perform a quantum chemical calculation to study it. But here we want to study this with easier, phenomenological method. It is known that HF has a large quadrupole moment of 1.875 au. Is it possible that this quadrupole moment leads to nonlinear geometry? Let’s examine this by cg-optimizer.

The use of cg-optimizer is very similar to rigidmol-optimizer and lego. We need a *.inp, *.cluster, and *.cg (not *.xyz). These files can be found in testfiles/coarsegrained/qm.

The configuration file q.inp:

```
q.cluster    # cluster file name
20            # population size
100           # maximal generations
3             # scout limit
4.00000000    # amplitude
q             # save optimized configuration to .xyz and .gjf
30            # number of LMs to be saved
```

The cluster file q.cluster:

```
1
q1.cg 2
* 10
```

Here, q1.cg is the coarse-grained structure file:
CHAPTER 6. COARSE-GRAINED PARTICLE CLUSTERS

Figure 6.1.1: Illustration of interaction sites.

<table>
<thead>
<tr>
<th></th>
<th>dipole</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>N</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>O</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIP4P q</td>
<td>epsilon (kJ/mol) sigma (AA)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LJC</td>
<td>0.0</td>
<td>1.0</td>
<td>2.4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DM</td>
<td>0.72</td>
<td>0.0</td>
<td>0.0</td>
<td>1.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>QM</td>
<td>1.875</td>
<td>0.0</td>
<td>0.0</td>
<td>1.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
6.2. EXAMPLE 1: WHY (HF)$_2$ IS NONLINEAR?

This is very similar to the force field parameter file *.xyz for rigidmol-optimizer. The atom coordinates are those of the interaction sites. For a more elegant model, usually the dipole moment and quadrupole moment should have a displacement, but here we make their centers identical. Of course we need the dipole moment $\text{DM}$ and quadrupole moment $\text{QM}$. To simulate the Pauli repulsion, we add a LJC site. The grammar is:

1. **LJC 0.0 1.0 2.4**. A Lennard-Jones–Coulomb particle with $q = 0.0, \epsilon = 1.0, \sigma = 2.4$.  
2. **DM 0.72 0.0 0.0 1.0** A dipole moment with $d = 0.72, \mathbf{n} = (0.0, 0.0, 1.0)$.  
3. **QM 1.875 0.0 0.0 1.0** A quadrupole moment with $Q = 1.875, \mathbf{n} = (0.0, 0.0, 1.0)$.

Note that in the input of $\mathbf{n}$ need not to be a normalized vector. It will be normalized internally. Also, for the coordinates, the element name C, N and O are arbitrary, here we use different names to distinguish the LJC, DM and QM.

Now we can search the GM of this q1 dimer:

```
cg-optimizer q.inp > q.out
```

The GM can be found in q-OPT.cluster and is shown in Figure 6.2.1. It is indeed a nonlinear geometry. Of course, with optimized parameters, it is possible to reproduce the accurate geometry. Thus, it may be inferred that the quadrupole moment of HF leads to the nonlinear geometry of (HF)$_2$. In fact, some researchers argue that the description of chemical bonding effects of a hydrogen bond can be understood in a purely classical fashion, provided that the description of the electrostatics is sufficiently complete and accurate.

You can also try to see the GM of (q1)$_{12}$. Just change q1.cg 2 in q.cluster to q1.cg 12. The result is shown in Figure 6.2.2.
6.3 Example 2: A Charge and Dipole Moments

Assume we have a positive charge and a dipole moment, then the minimum is a head-to-tail structure. But what will it be if we have several dipole moments? Let’s check it by cg-optimizer. Our model is: a positive charge and 14 dipole moments. The corresponding input files are in testfiles/coarsegrained/charge-dipole.

We first build a configuration file cd.inp:

```
cd.cluster # cluster file name
10 # population size
50 # maximal generations
3 # scout limit
4.00000000 # amplitude
cq35 # save optimized configuration to .xyz and .gjf
30 # number of LMs to be saved
```

then a cluster file cd.cluster:

```
2
c.cg 1
d.cg 14
* 10
```

The structure of the positive charge is c.cg:

```
1
charge
0 0.0 0.0 0.0
TIP4P q epsilon (kJ/mol) sigma (AA)
LJC +2.0 1.0 2.4
```
6.4. EXAMPLE 3: PY CLUSTERS

The structure of the dipole moment is \texttt{d.cg}:

\begin{verbatim}
2
dipole
S 0.0 0.0 0.0
N 0.0 0.0 0.0
TIP4P q  epsilon (kJ/mol) sigma (AA)
LJC 0.0 1.0 2.4
DM 3.5 0.0 0.0 1.0
\end{verbatim}

For both the charge and dipole moment, we have made their LJ parameters nonzero in order to simulate the Pauli repulsion and prevent the system from collapsing into negative infinity.

Now we can run the simulation:

\texttt{cg-optimizer cd.inp > cd35.out}

Now you can find the structure of the GM in \texttt{cd35-OPT.cluster} and in Figure 6.3.1. For the GM, the dipole moments form a ring in head-to-tail pose, and the charge lies at an off-center position.

You can change the dipole moment to some smaller values to see what will happen. Just change \texttt{DM 3.5} in \texttt{d.cg} to \texttt{DM 1.5} and \texttt{DM 0.5} and run \texttt{cg-optimizer} again. Interestingly, the GM of \texttt{d = 1.5} is again a straight line with the charge at the starting point. For \texttt{d = 0.5}, a star-shape GM is observed. See Figure 6.3.1.

![Figure 6.3.1: The GM of LJC-(DM)14 with d = 0.5, 1.5 and 3.5.](image)

6.4 Example 3: PY Clusters

Now we want to search the GM of 13 PY particles. As we mentioned, the PY interaction is an “anisotropic” Lennard-Jones potential, leading to very interesting geometries. The corresponding input files are in testfiles/coarsegrained/py.

First, we prepare the configuration \texttt{py.inp} and cluster \texttt{py.cluster} files:

\begin{verbatim}
py.cluster  # cluster file name
10          # population size
\end{verbatim}
CHAPTER 6. COARSE-GRAINED PARTICLE CLUSTERS

<table>
<thead>
<tr>
<th>20</th>
<th># maximal generations</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td># scout limit</td>
</tr>
<tr>
<td>4.00000000</td>
<td># amplitude</td>
</tr>
<tr>
<td>py-prolate</td>
<td># save optimized configuration to .xyz and .gjf</td>
</tr>
<tr>
<td>30</td>
<td># number of LMs to be saved</td>
</tr>
</tbody>
</table>

```
1 py.cg 13
* 10
```

For the PY particle, we prepare the `py.cg`:

```
1
PY
S 0 0 0
q epsilon (kJ/mol) sigma (AA)
PY 1.0 1.0 1.5 0.5 0.5 1.407 0.5 0.5 1.0 0.0 0.0 0.0 1.0 0.0
```

The grammar of PY is:

`PY 1.0 1.0 1.5 0.5 0.5 1.407 0.5 0.5 1.0 0.0 0.0 0.0 1.0 0.0`:

\[
\begin{align*}
\epsilon &= 1.0; \\
\sigma &= 1.0; \\
a_1^R &= 1.5; \\
a_2^R &= 0.5; \\
a_3^R &= 0.5; \\
b_1^R &= 1.407; \\
b_2^R &= 0.5; \\
b_3^R &= 0.5; \\
\mathbf{u}_1 &= (1.0,0.0,0.0); \\
\mathbf{u}_2 &= (0.0,1.0,0.0) \\
\end{align*}
\]

You do not need to input \( \mathbf{u}_3 \), it will be calculated by ABCluster by \( \mathbf{u}_3 = \mathbf{u}_1 \times \mathbf{u}_2 \).

Note that here one axis is much longer than the other two ones (1.5 vs 0.5, 0.5), thus the PY is a prolate ellipsoid.

OK, now we run `cg-optimizer`:

```
cg-optimizer py.inp > py-prolate.out
```

We can also try to find the GM of an oblate ellipsoid PY particle (that is, one axis is much shorter than the other two ones), just need to change `py.cg`:

```
1
PY
S 0.0 0.0 0.0
q epsilon (kJ/mol) sigma (AA)
PY 1.0 1.0 0.15 0.5 0.5 0.06 0.5 0.5 1.0 0.0 0.0 0.0 1.0 0.0
```

and also change `py-prolate` in `py.inp` to `py-oblate`. Run the search:

```
cg-optimizer py.inp > py-oblate.out
```

Now you can find the GMs in `py-prolate-OPT.cluster` and `py-oblate-OPT.cluster`. See Figure 6.4.1. You may note that PY clusters can form helix structures, being very anisotropic and even chiral!

6.5 Example 4: A Simple Model of Amphiphilic Molecule

In previous sections, we studied interaction sites like polar DM, QM and nonpolar PY. If we combine them together, we could build a model of amphiphilic molecule. A simple model is to attach a PY with two DMs on each side, e.g. the `*.cg` can be written in this way:
6.5. EXAMPLE 4: A SIMPLE MODEL OF AMPHIPHILIC MOLECULE

Figure 6.4.1: The GM of prolate and oblate (PY)\textsubscript{13}.

\begin{verbatim}
5
AMPH
P  0.0  0.0  0.0
O  +3.0 0.0  0.0
O  +3.0 0.0  0.0
N  -3.0 0.0  0.0
N  -3.0 0.0  0.0
AMPH1 epsilon (kJ/mol) sigma (AA)
PY  1.0  10.0  2.8  0.5  0.5  2.407  0.5  0.5  1.0  0.0  0.0  0.0  1.0  0.0
   0.0
LJC  0.0  1.0  1.4
DM  0.1  0.0  0.0  1.0
LJC  0.0  1.0  1.4
DM  0.4  0.0  0.0  1.0
\end{verbatim}

This is shown in Figure 6.5.1. As done in previous sections, we can search the GM of its polymers. The GM of its pentamer is shown here. You can also find the relevant files in testfiles/coarsegrained/amph. More realistic models can be built by adding more interaction sites and fitting parameters with experiments.

Figure 6.5.1: A simple model of amphiphilic molecule and the GM of its pentamer.
6.6 Visualization of Coarse-Grained Particles

Unlike the examples in previous chapters, where all clusters involved are well-defined chemical objects thus one can visualize them with numerous software. The coarse-grained particles defined in ABCluster cannot be directly interpreted as molecules, thus the author wrote a small software called \texttt{cgviewer} to visualize the *.cluster generated by ABCluster.

\texttt{cgviewer} is not in the distribution of ABCluster. It has to be downloaded from:

\url{http://www.zhjun-sci.com/software-abcluster-download.php}

\texttt{cgviewer} is very easy to use, just use \texttt{File→Open} to select the *.cluster, then it will be displayed in the OpenGL window. You can use \texttt{Render} to turn on or off the coordinate system and lights, or change background color. As explained in Section , the LJC, DM, QM and PY particles are visualized as ball, arrow, disk, and ellipsoid. The color is determined by its “element name”. \texttt{cgviewer} is still in 0.1 version, the author will try to improve it in future.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{cgviewer_snaps.png}
\caption{Snapshots of \texttt{cgviewer}.}
\end{figure}
Appendix A

Version History

- Version 2.0 (Released on: Aug. 1, 2018):
  1. For `rigidmol-optimizer`, ABCluster can study the molecular clusters in an external electric field.
  2. For `isomer` and `lego`, the algorithm to generate initial guess (thus the search efficiency) has been improved significantly (For example, clashes removing, coarse optimization)!
  3. `isomer` and `lego` has been parallelized for both single and many nodes!
  4. Official support for `isomer` and `lego` with DMol3 is provided (Courtesy of Mr. Kai Wang).

- Version 1.5.1 (Released on: Dec. 18, 2017):
  1. Official support for `isomer` and `lego` with ORCA, NWChem, and xTB-GFN are provided.
  2. The evaluation codes have been optimized.

- Version 1.5 (Released on: Dec. 18, 2016):
  1. A component `cg-optimizer` is added to support the anistropic, coarse-grained particles, including electric multipole interaction and Paramonov–Yaliraki potential.
  2. The component `rigidmol-optimizer` supports 1D, 2D and 3D periodic boundary conditions.
  3. New modified Sutton–Chen \( (J. \ Phys. \ Chem. \ Solids, 2015, 82, 67–75) \) and extended Lennard-Jones potential are supported.
  4. A new keyword `surface` is added to `lego` for better supporting surface absorption.
  5. The output of `isomer` and `lego` are optimized.

- Version 1.4 (Released on: Apr. 5, 2016):
  1. The new component `lego` has been added to support to search by third-party programs for both molecular and atomic clusters.
  2. New Tersoff potential is supported.

- Version 1.3 (Released on: Jan. 22, 2016): New algorithm has been introduced. Now the efficiency of multi-component atomic clusters (e.g. \( \text{Ag}_{14}\text{Cu}_{24} \)) has been improved by 100 times!
• Version 1.2 (Released on: Nov. 6, 2015): The interfaces for third-party programs are available! Now ABCluster can do the global optimization with any quantum chemistry programs!

• Version 1.1 (Released on: Oct. 19, 2015): The optimization of Gupta and Sutton–Chen potential is significantly improved.

• Version 1.0 (Released on: Oct. 1, 2015): The first released version.
Bibliography


