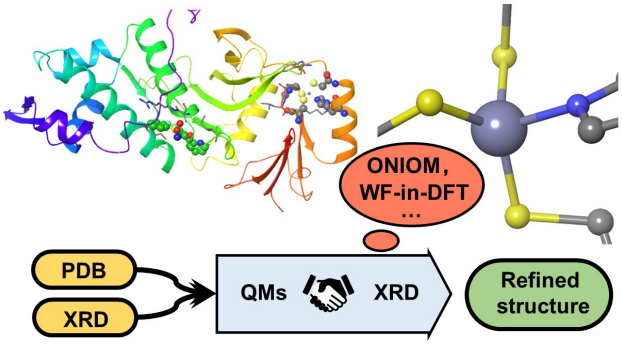
User's Manual for DL_FIND based Protein Quantum Refinement Program

Zeyin Yan

June 1, 2021

Biomolecules with metal ion(s) (e.g., metalloproteins) play many important biological roles. However, accurate structural determination of metalloproteins (particularly containing transition metal ion(s)) is challenging due to their complicated electronic structure, complex bonding of metal ions and tremendous conformations in biomolecules. Quantum refinement which was proposed to combine crystallographic data with computational chemistry methods by several groups can improve the local structures of some proteins. This code is an interface to realise quantum refinement method combining several multiscale computational schemes with experimental (X-ray diffraction) information for metalloproteins.



Link: `https://github.com/oscarchung-lab/ONIOM_QR`

1 Introduction

This interface is developed based on DL-FIND open-source geometry optimisation library. This program combines the Gaussian (for ONIOM part) and CNS (experimental part) as shown Figure 1. Other QM/MM, QM and SE methods using different *ab-initio* codes are also available through user-defined scripts.

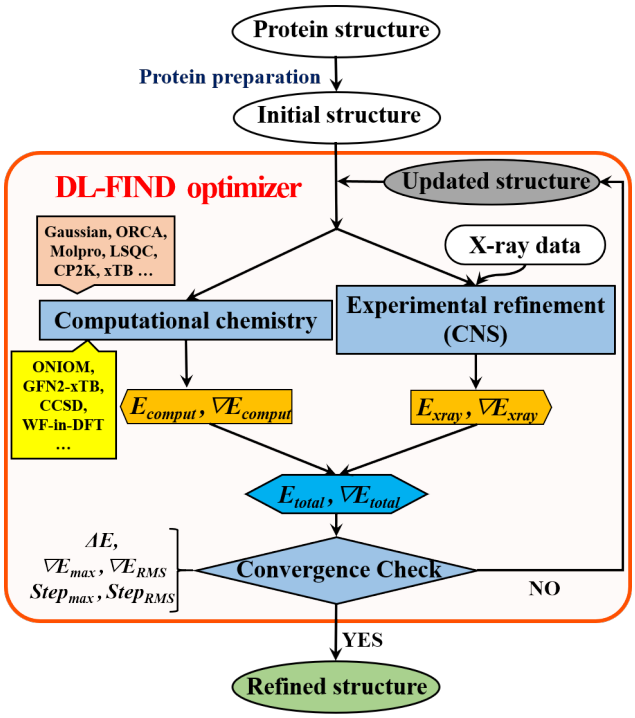


Figure 1: DL-GAUCNS Quantum Refinement process

1.1 Installation

DL-FIND open-source geometry optimisation library should be downloaded first (`https://www.chemshell.org/dl-find`). Then, *main_ONIOM_QR.f90 Fortran* script is copied to replace the *main.f90* file in DL-FIND folder. Adjust parameter in the *Makefile* according to your *Fortran* and complier the code to obtain the executable file *find.x*.
Download and install CNS(`http://cns-online.org/v1.3/`, 1.3 version is preferred, other version is not guaranteed).
Other *ab-initio* codes (eg: Gaussian, ORCA etc) should be installed before refinement.

Modified CNS files
To extract the force and energy data from CNS refinement, the CNS input files are modified (according from Comqun code of Prof. Ryde `http://signe.teokem.lu.se/~ulf/Methods/comqum_x.html`). These file are offered in CNS folder, which should be copied to the dl_gaucns location($DF/CNS/).

1.2 Scripts and Programs

1.2.1 Programs

sf-convert

Usage: `https://sw-tools.rcsb.org/apps/SF-CONVERT/doc/V1-2-04/documentation.html`

Usage: *sf-convert -i mmCIF -o CNS -sf xxx.cif*

Normally, the structure factors downloaded from the PDB website are in format mmCIF, which is not compatible with CNS.

check_link
Usage: *check_link xxx.gjf*
The program is used to check the ONIOM link and relaxed information. A new gjf file with corrected information. Since if the high level atom are relaxed, the link atoms in lower level should be relaxed too. If not Gaussian will report error.

force_grad

Usage: *check_link filename num_atom*

The program is used to transfer gradient and force (sign of values).

update_gjf

Usage: *update_gjf origin.gjf traget.gjf*

The program is used to update the atom coordinates in target.gjf from origin.gjf.

gau2mpxyz

Usage: *gau2mpxyz Gauinput GauMP.xyz*

The program is transfer Gaussian external input file(Gauinput, format: `https://gaussian.com/external/`) to Molpro input xyz (GauMP.xyz) file. This program is called by *call_molpro.sh*.

Multi-center ONIOM scheme:
Two methods are offered to define fragments of high layer in ONIOM scheme. Both methods generate a file called center_gjf.bak, which contains all the gjf files of fragments for checking the fragmentations information.

The first one is defined by the center atoms of each fragments, the other atoms are separated to fragments according to the minimal distance to the center atoms. This method is simple and works well for the case where fragments are far away to each other. If the fragments are close, it may cause problem.

***GauCenters_list:* external for Gaussian**

Usage: *External='GauCenters_list parafile'*

parafile is a text file to define atom lists and other information.

```
2                       #number of fragments
1-17,36                 #atom list (including link atoms) of fragment 1
18-35                   #atom list (including link atoms) of fragment 2
head1.txt tail1.txt    #head and tail files for Gaussian computation of fragment 1
head2.txt tail2.txt    #head and tail files for Gaussian computation of fragment 2
```

Or if we need to keep the lower-level interaction contribution

```
2                       #number of fragments
1-17,36                 #atom list (including link atoms) of fragment 1
18-35                   #atom list (including link atoms) of fragment 2
head1.txt tail11.txt    #head and tail files for Gaussian computation of fragment 1
head2.txt tail12.txt    #head and tail files for Gaussian computation of fragment 2
head_low.txt tail_low.txt    #head and tail files for lower-level Gaussian computation of Model
head1_low.txt tail1_low.txt     #head and tail files for lower-level Gaussian computation of fragment 1
head2_low.txt tail2_low.txt     #head and tail files for lower-level Gaussian computation of fragment 2
```

Here, the atoms list is different to the label in gjf file of whole system. Thus, the tool to obatin the correct atomic label is implemented in ***GauCenters_list***.

Usage: *External='GauCenters_list'*

Call the external program GauCenters_list in Gaussian without extra parameter, the job will be terminated (error) with generating **testtmp.gjf** file, which are the high layer with link atoms. Therefore, users can use the *Atom selection* tools in GaussianView to obtain the atom list string.

***call_lsqc:* external for Gaussian**

Usage: *External='call_lsqc lsqc.gjf'*

This program is call the fragmentation program (open-source by Li shuhua group, Nanjing University, `https://itcc.nju.edu.cn/lsqc/`). *lsqc.gjf* is LSQC input file, which is very similar to Gaussian gjf file.

All the source *Fortran* code are put in the source folder.

1.2.2 Scripts

dlg_cns:

Usage: *dlg_cns xxx.pdb*

Copy the modified CNS files to working directory, and fill crystal information into *sedfile* from the xxx.pdb. Parts of parameters in *sedfile* still need to be filled manually such as: QR parameters, the parameter and topology files for the ligands etc.

dlg_pre:

Usage: *dlg_pre*

Update information in modified CNS files according the *sedfile*.
Run the *generate.inp* and *make_cv.inp* CNS jobs. The parameter and topology files of ligands and structure factor files should be copied to the work directory before running this script.

dlg_clean:

Usage: *dlg_clean*

Clean up the useless files.

An example is shown in Figure 2

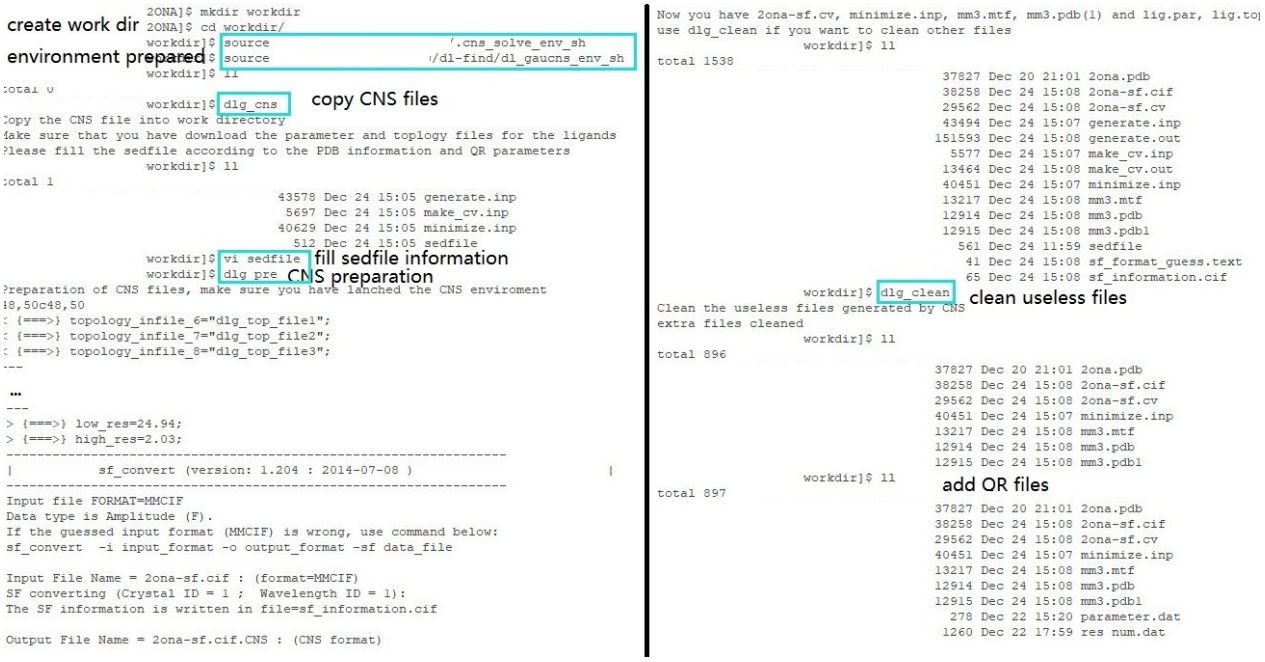


Figure 2: An example of using script tools

***call_molpro.sh:* external for Gaussian**

Usage: *External='call_molpro.sh MPinput.com ncpu'*

This script is a Gaussian external interface calling Molpro to conduct WF-in-DFT computation. The program *gau2mpxyz* is needed.

1.3 PATH

Set up the CNS environment (cns_solve_env_sh in the CNS directory) before using QR interface.

```
# >>>>>> Important: define the location of the CNSsolve directory <<<<<<
#
# CHANGE THE NEXT LINE TO POINT TO THE LOCATION OF THE CNSsolve DIRECTORY

CNS_SOLVE=/xxx/xxx/xxx/xxx #path of the CNS
#
# ============================================================================
```

Edit the *dl_gaucns_env_sh* environment file:

```
#DF-GAUCNS program location
export DF=/xxx/.../df_gaucns

#Gaussian command according to your gaussian version
```

```
export Gauexe="g09" # g16 or g09
#path of Gaussian
export g09root=/share/apps/gaussian
. $g09root/g09/bsd/g09.profile
# or Gaussian16
#export Gauexe="g16"
#export g16root=/share/apps/gaussian
#. $g16root/g16/bsd/g16.profile

#some scripts for QR including preparation and Gaussian external interface
chmod +x $DF/scripts/*

export PATH=$PATH:$DF/scripts

#CNS environment
source /xxx/.../cns_solve_1.3/cns_solve_env_sh

#other QM code environment if needed

#xtb
source /share/home/yanz/Apps/xtb/Config_xtb_env.bash

#LSQC
export lsroot=/share/home/yanz/Apps/LSQC/lsqc-2.4
export PATH=$lsroot/bin:$PATH
export LD_LIBRARY_PATH=$lsroot/lib:$LD_LIBRARY_PATH
export CIM_BASDIR=$lsroot/basis
export KMP_STACKSIZE=1G
export LD_LIBRARY_PATH=/share/home/yanz/Apps/LSQC/cint_and_xc/lib:$LD_LIBRARY_PATH
export PYTHONPATH=/share/home/yanz/Apps/LSQC/software/pyscf-master:$PYTHONPATH

source /share/env/python-2.7.16.env

#....
```

Then load the environment by:

```
source /xxx/.../dl_gaucns/dl_gaucns_env_sh
```

2 Work Flow

To conduct the Quantum Refinement (QR) using DF-GAUCNS program, the work flow is presented as following:

PDB files

The data needed for QR is PDB structure (.pdb) and structure factor data (.cif), which can be downloaded from the database (https://www.rcsb.org/) or other where.
If there are ligands, then the parameter and topology files are needed (CNS part), which can be downloaded form the database (http://xray.bmc.uu.se/hicup/) or other where.

Preparation

For the QM part, hydrogen should be added, using PDB2PQR, Schrodigner etc. Using the PDB with H geometry to define a QM single point energy computation file with gradient output.
For the CNS part, there are several steps:

1 copy the CNS model files to the work directory *cp* $\$CNS\_DIR/script/*$ $\$WORK\_DIR$;

Quantum refinement with or without B factor depends on if the *bindividual.inp* is prepared in the working directory.

2 copy parameter and topology files to the work directory if needed

3 fill the *sedfile* PDB information (PDB file);

4 change the CNS inp files for the PDB case by a script *sh dlg_sedscript.sh*

5 generate CNS geometry files mm3.mtf and mm3.pdb *cns* < *generate.inp* > *generate.out*

6 mm3.pdb1 can be generated using script to give more accurate result during refinement

```
head -n 3 mm3.pdb | sed s/mm3.pdb/mm3.pdbs/ > mm3.pdb1
sed -n '4,/^END/p' mm3.pdb | sed '$d' | sed 's/[-]/ /g' |sed 's/[0-9]\{1\}\.[0-9]\{3\}/0.000/g' &
|sed 's/[0-9]\{2\}\.[0-9]\{3\}/ 0.000/g' | sed 's/[0-9]\{3\}\.[0-9]\{3\}/  0.000/g' >> mm3.pdb1
echo 'END' >> mm3.pdb1
```

6 transfer the structure factor file by *sf_convert* to a CNS compatible format, then treated by CNS using *cns* < *make_cv.inp* > *make_cv.out*

Now, you should get dlg_relex_outfile, minimize.inp, mm3.mtf, mm3.pdb, mm3.pdb1, and lig.par, lig.top if there is. These file will be used for the QR.

QR Definition

RESFILE can be defined according to **RESFILE** explication (recommended).
MAPFILE can be defined according to **MAPFILE** explication (can be generated auto by program for the first time, however, if you restart the QR when the QM and CNS geometry are different, the file saved by program the first time will be useful and highly recommended).
Parameter file should be defined according to **Keywords** section.

Begin the QR

After you define the environment for the QM and CNS environment, the QR can be launched using *find.exe parameter.dat*

3 Keywords

3.1 Parameters

GAUFILE - Gaussian input file
string

inputfile gauinput for the Gaussian single point energy input file name (with extra keywords **nosymm force punch=(coord,derivatives)**)

The GAUFILE is for a Gaussian computation, no matter what kinds of computation (QM, MM, Semi-empirical or ONIOM), it depends on the system and your choice. It is a normal single point energy computation with extra gradient output using keywords **nosymm force punch=(coord,derivatives)**. The energy value is extracted from output file, and gradients from *fort.7* file.
Since, the interface using Fortran language, the format of atom block is strict: The GaussianView is recommended to prepare the atom block in gjf file. Three case are possible:

(1X,A15,3F14.8) line length is less than 60 (normally 58)

As shown (a) in Figure 3, the simple Gaussian input file contains only element and Cartesian coordination information.

1X: one space; **A15**: elements; **3F14.8**: coordinations including x, y, z;

(1X,A18,3F14.8) line length is less than 63 (normally 61)

As shown (b) in Figure 3, the Gaussian input file contains element, Cartesian coordination and freeze information.

1X: one space; **A18**: elements **A15** and freeze information **A3**; **3F14.8**: coordinations including x, y, z;

(1X,A18,3F14.8,1X,A1) line length is larger than 62 (normally 63 or larger with link atom information)

As shown (c) in Figure 3, the Gaussian input file contains element, Cartesian coordination, freeze and ONIOM information.

1X: one space; **A18**: elements **A15** and freeze information **A3**; **3F14.8**: coordinations including x, y, z; **1X,A1**: ONIOM layer information and link atom

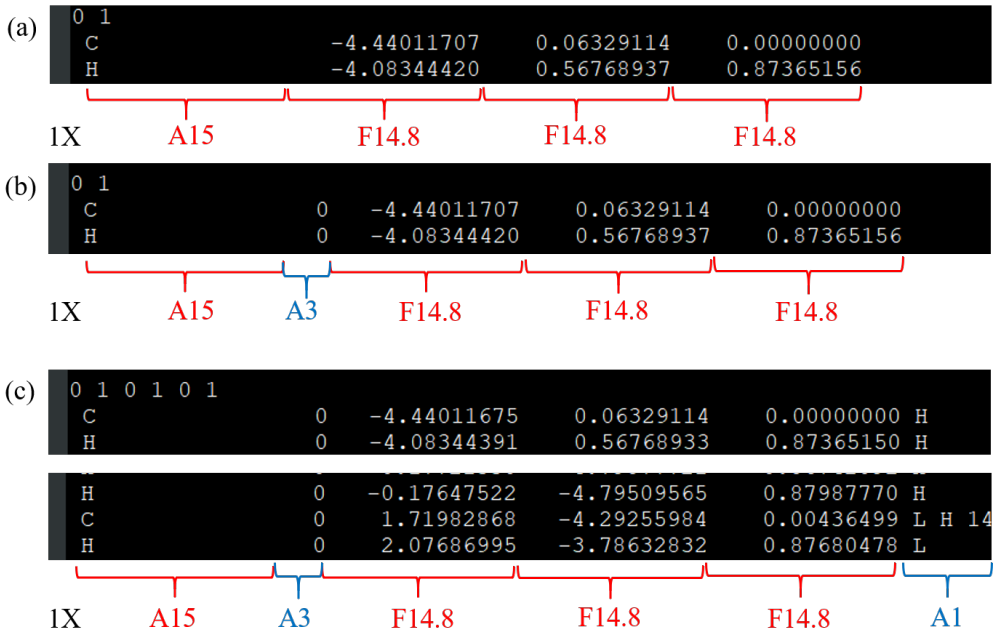


Figure 3: Three cases of atomic block format in Gaussian input file: (a) **(1X,A15,3F14.8)**, only element and coordination information; (b) **(1X,A18,3F14.8)**, element, coordination and freeze information; (c) **(1X,A18,3F14.8,1X,A1)**, element, coordination, freeze and ONIOM information.

METHOD - Refinement methods
int

0 for external QM program + CNS refinement

1 for GAUSSIAN optimization only

2 for GAUSSIAN + CNS refinement

If **0** is chosen, there are three files should be prepared: **coord.xyz**, **relaxed.dat** and **external.sh** (examples of ORCA and LSQC will be detailed):

coord.xyz chemical file format specifies the molecule geometry by giving the number of atoms with Cartesian coordinates that will be read on the first line, a comment on the second, and the lines of atomic coordinates in the following lines:

```
<number of atoms>
comment line
<element> <X> <Y> <Z> #Angstroms
...
```

relaxed.dat total number (**NA**) of relaxed atoms on the first line, a list of integer of relaxed atomic numbers in the following lines:

```
NA
i
.
.
.
j
.
.
.
```

external.sh a script that realizes three functions: transfer geometry information from **coord.xyz** to external QM program, call external QM program, output a file **QM_grad.dat** (a.u.) containing QM energy on the first line and gradient results in the following **HNAT** lines:

```
<QM energy>
<gradX1> <gradY1> <gradZ1>
...
<gradXn> <gradYn> <gradZn>
```

NAT - Number of atoms in PDB file
int

HNAT - Number of system atoms with H atoms
int

MAPPING - Mapping the PDB and G09 input file atoms
int

0 n – n, the H in the end of the file

1 read from a map file

2 auto mapping based on the coordinates, the result will be saved as *mapfile.dat*, if there is different between two coordinates, error information will be written and program stops.

MAPFILE - file name of mapping file
string

a list of NAT integer to explain the position of ith atom in the G09 input file

RESNUM - protein residue information, useful for internal coordinates
int

0 all atoms are treated in the same residue

1 read from a residue information file

RESFILE - residue information file
string

list of HNAT integer to indicate the residue number of atoms.
It can be extracted from the PDB file with H atoms added

MAXENE - maximum number of E& G evaluations
int

MAXSTEP - maximum length of the step in internals
float

LBFGS_MEM - number of steps in LBFGS memory
int

TOLERANCE - convergence criterion
float float

main convergence criterion (max grad comp.)

convergence criterion on energy change

MAXCYCLE - maximum number of cycles
int

NCON - defined as default 0 0
int int

number of constraints

number of user provided connections

IOPT - Type of optimisation algorithm
int

0 Steepest descent

1 Conjugate gradient following Polak–Ribière (with automatic restarts based on the criterion by Powell and Beale) that is not coded properly ...

2 Conjugate gradient following Polak–Ribière with CG restart every 10 steps (hardcoded at the moment)

3 L-BFGS

9 Test delta for finite-difference in gradients (19 energy and gradient evaluations)

10 P-RFO A switching mechanism for the mode to be followed is included, but does not seem to help in any of the cases I tried sofar.

11 Just calculate the Hessian and do a thermal analysis (harmonic approximation for entropy, ...)

12 Calculate the Hessians of all images and the qTS rate (if inithessian=5: just the rate, read the Hessians)

13 Rate without tunneling (only with Wigner correction)

20 Newton–Raphson/quasi-Newton

30 Damped dynamics using the variables timestep, fric0, fricfac, and fricp. The frictions are defined that 0 corresponds to free (undamped) dynamics, and 1 corresponds to steepest descent.

51 Random (stochastic) search, using the variables po_pop_size, po_radius, po_contraction, po_tolerance_r, po_tolerance_g, po_distribution, po_maxcycle, po_scalefac.

52 Genetic algorithm , using the variables po_pop_size, po_radius, po_tolerance_g, po_maxcycle, po_init_pop_size, po_reset, po_mutation_rate, po_death_rate, po_nsave

ILINE - Type of line search or trust radius
int

0 simple scaling of the proposed step, taking maxstep into account

1 Trust radius based on energy as acceptance criterion (recommended for L-BFGS optimisation)

2 Trust radius based on gradient as acceptance criterion (recommended for CG optimisation)

3 Hard-core line search. Does not work at the moment...

INITHESSIAN - Type of initial Hessian
int

0 Calculate externally using `dlf_get_hessian`. Defaults to two point finite difference if an external Hessian is unavailable.

1 Build by one point finite difference of the gradient

2 Build by two point finite difference of the gradient

3 Build a diagonal Hessian with a single one point finite difference

4 Set the Hessian to be an identity matrix

5 Only for instanton calculations: read the Hessian from `qts_hessian.txt`

6 Only for instanton calculations: read the Hessian from files for each image`qts_hessian_imageX.txt`

UPDATE - Hessian update mechanism
int

0 No update. Always recalculate the Hessian

1 Powell update

2 Bofill update

3 BFGS update

IMICROITER - Microiterative optimisation
int

0 Standard (non-microiterative) optimisation

1 Microiterative optimisation

ICOORD - Type of coordinate system
int

"unit place" means `icoord` modulo 10

0–9 The whole system is to be treated as one image

Unit place 0 Cartesians

Unit place 1 HDLC - internals

Unit place 2 HDLC - TC

Unit place 3 DLC - internals

Unit place 4 DLC - TC

Unit place 5 Mass-weighted Cartesians ($\sqrt{m}x$) – will be deleted:

PRINT - Print level
int int

0 no printout

2 print something

4 be verbose

6 debug

4 Examples

Examples (apply on 6NB9 as an example) are given to explain how to use QR_ONIOM code.
The ONIOM based example are given using 1-center, 2-center strategies. The test using Gaussian as a full QM computation are also given. Moreover, a test of two-center Gaussian computation is given using *GauCenters_dis* and *GauCenters_list* are also given.

4.1 External Examples

Three examples of external method are introduced here to explain how to use QR_ONIOM code to do QR with different QM programs. As presented before, the **coord.xyz**, **relaxed.dat** and **external.sh** files should be prepared, and maybe extra keyword need to be added in the QM input file. Among them, **coord.xyz** and **relaxed.dat** have been well defined in the **METHOD** keyword, and here the **external.sh** will be detailed. It is a script that realizes three functions: transfer geometry information from **coord.xyz** to external QM program, call external QM program, output a file **QM_grad.dat** (a.u.) containing QM energy on the first line and gradient results in the following **HNAT** lines

ORCA case

ORCA is an ab initio quantum chemistry program package that contains modern electronic structure methods including density functional theory, many-body perturbation, coupled cluster, multireference methods, and semi-empirical quantum chemistry methods. Its main field of application is larger molecules, transition metal complexes, and their spectroscopic properties. ORCA is developed in the research group of Frank Neese. The free version is available only for academic use at academic institutions. More information can be found in the website `https://orcaforum.kofo.mpg.de/app.php/portal`

```
#!/bin/bash
#orca as QM program
inpf='6nb9.inp'
#Update external QM program input file (geometry info from coord.xyz)
jobn=`echo $inpf | sed 's/.inp//g'`
atomn=`head -n 1 coord.xyz | tr -cd "[0-9]"`
a=`grep -rin xyz $inpf | awk -F ":" '{print $1}'`
b=$(($atomn+2))

head -n $a $inpf > temp.inp
sed -n '3,'$b'p' coord.xyz >> temp.inp
echo '*' >> temp.inp
mv temp.inp $inpf

cp $PBS_O_WORKDIR/*.inp $orca_dir/ 2>/dev/null
cp $PBS_O_WORKDIR/*.gbw $orca_dir/ 2>/dev/null
cp $PBS_O_WORKDIR/*.xyz $orca_dir/ 2>/dev/null

cd $orca_dir
ls -l > $PBS_O_WORKDIR/orca.log
echo $orca_dir >> $PBS_O_WORKDIR/orca.log
#call external QM program



$ORCA_BIN/orca $jobn.inp >> $PBS_O_WORKDIR/orca.log
```

```
cp $orca_dir/*.gbw $PBS_O_WORKDIR/ 2>/dev/null
cp $orca_dir/*.xyz $PBS_O_WORKDIR/ 2>/dev/null
cp $orca_dir/*.engrad $PBS_O_WORKDIR/ 2>/dev/null

cd $PBS_O_WORKDIR
#output the energy and gradient results in $jobn.engrad to the QM_grad.dat file

sed -n '8p' $jobn.engrad > QM_grad.dat
sed -n '12,/^#/p' $jobn.engrad | sed '$d' | awk 'ORS=NR%3?"\t":"\n"' >> QM_grad.dat
```

For the ORCA input file, the keyword **EnGrad** should be added to output the **$name.engrad** gradient files.

LSQC case

Low Scaling Quantum Chemistry (LSQC) program (No. 2006SR09617) is a quantum chemistry package for low scaling or linear scaling electronic structure calculations, developed by the research group of Professor Shuhua Li in Nanjing University. The package can be used for calculating the energy, geometry, frequency, excited state and other properties of molecules and molecular crystals with ab initio (HF and post HF) and density function theory (DFT) methods. More information can be found in the website `http://itcc.nju.edu.cn/lsqc`

```
#!/bin/bash
#lsqc as QM program
#define the parameter of the gjf file
#atom=224
title=10
#xyz='2ona.xyz'
gjf="6nb9.gjf"
atom=‘head -n 1 coord.xyz | tr -cd "[0-9]" ‘
#calculate number
a=$(($title+1))
b=$(($atom+$a))
c=$(($atom+3))
dir=${gjf%.*}
#Update external QM program input file (geometry info from coord.xyz)
if [ ! -d "$dir" ]; then
    head -n $title $gjf > temp.gjf
else
    cp $dir/$dir/$dir.cha .
    cp $dir/$dir/$dir.frg .
    cp $dir/$dir.gjf .
    rm -rf $dir
    head -n $title $gjf | sed s/'\scharge=[a-zA-Z]*'/' charge=read'/g | sed s/'\sfrag=[a-zA-Z]*'/' frag=
fi
sed -n '3,'$c'p' coord.xyz >> temp.gjf
sed -n $b',/^$/p' $gjf >> temp.gjf
mv temp.gjf $gjf

#call external QM program
```

```
lsqc $gjf > $dir.out

#outputthe energy and gradient results in 2ona.engrad to the QM_grad.dat file
cp $dir/$dir/$dir.force .
cp $dir/$dir/$dir.out .
head -n 1 $dir.out | awk '{print $4}' > QM_grad.dat
force_grad $dir.force $atom
cat $dir.force >> QM_grad.dat
```

For the LSQC input file (similar with Gaussian input file), keyword **force** should be added to generate the **$dir.force** file.
To conduct the LSQC MPI parallel computation, *unitcore* = 40 value in file *.../LSQC/bin/lsqc_parallel.sh* (line 19) should be changed to the number of CPUs per node, and the *hosts* file should be generated before the computation:
For IBM Platform Load Sharing Facility (LSF), use command:

```
awk ' !x[$0]++' $LSB_DJOB_HOSTFILE > hosts
```

For Portable Batch Systems (PBS), use command:

```
cat $PBS_NODEFILE | uniq > hosts
```

GFN-xTB case

GFN-xTB is a semi-empirical computational package based on Tight-binding based simplified Tamm-Dancoff approximation(sTDA-xTB), developed by Grimme and co-workers for predicting Geometries, vibrational Frequencies and Noncovalent interactions ("x" stands for extensions in the AO basis set and the form of the Hamiltonian). More information can be found in the website `https://www.chemie.uni-bonn.de/pctc/mulliken-center/software/xtb`, and the package are free for academic use upon request.

```
#!/bin/bash
#xtb as QM program
#Update external QM program input file (geometry info from coord.xyz)
#coord.xyz can be directly used

#call external QM program
#xtb has been added to the PATH
xtb coord.xyz --chrg 1 --uhf 0 --grad > xtb.out

#output the energy and gradient results in 2ona.engrad to the QM_grad.dat file
atom=‘head -n 1 coord.xyz | tr -cd "[0-9]"‘

#echo $atom

a=$(($atom+3))
b=$(($atom+$atom+2))

#echo $a $b # gradient file will be written continuously the next time
#rm -f gradient
#tail -n $b gradient | grep 'SCF energy' | awk '{print $7}' > QM_grad.dat
```

```
#tail -n $b gradient | sed -n $a','$c'p' >> QM_grad.dat
grep 'SCF energy' gradient | awk '{print $7}' > QM_grad.dat
sed -n $a','$b'p' gradient >> QM_grad.dat


rm -f charges energy xtbrestart gradient hessian xtbout mol.xyz tmpxx vibspectrum
rm -f hessian xtb_normalmodes g98_canmode.out g98.out wbo xtbhess.coord .tmpxtbmodef
```

To use the parallel (**OPENMP**) computation of xTB:

```
export OMP_NUM_THREADS=N
export MKL_NUM_THREADS=N
export OMP_STACKSIZE=1000m
ulimit -s unlimited
```

is recommended, N is the number of CPUs on the same node.