OPENCAP-MD manual

Soubhik Mondal

February 5, 2025

Contents

1 Directory Structure of OPENCAP-MD

Below is the structured view of the OPENCAP-MD package.

```
OPENCAP-MD
|-- optimizer
|   |-- optimizer_MOLCAS.py
|   |-- optimizer_columbus.py
|-- utils
|   |-- read_mcdrtfl.py
|   |-- natorb_utils.py
|   |-- read_civfl_w_sym.py
|   |-- isotope_mass.py
|   |-- ParseFile_MOLCAS.py
|-- sharc-interface
|   |-- SHARC_MOLCAS_OPENCAP.py
|-- setup.py
```

Figure 1: Directory structure of OPENCAP-MD

2 Description of Files and Directories

Top-Level Directories

- **optimizer/**: Contains scripts for optimizing molecular structures using different quantum chemistry packages. Available interfaces are:

  `optimizer_MOLCAS.py`: Geometry optimizer for OpenMolcas

  `optimizer_columbus.py`: Geometry optimizer for Columbus

- **utils/**: Includes various utility functions for handling molecular data and parsing input/output files.
- **sharc-interface/**: Provides a MOLCAS-OPENCAP interface with SHARC AIMD package. Also used by `optimizer_MOLCAS.py`.
- **examples/**: Contains example cases for running OpenMolcas and Columbus based geometry optimizations.

2.1 OpenMolcas specific files

The files needed for OpenMolcas based optimization is structured as shown in 2.1. For reference see `examples/openmolcas` directory.

```
OPENCAP-MD
|-- examples
|    |-- openmolcas
|    |    |-- run-opencapmd.sh
|    |    |-- geom
|    |    |-- QM
|    |    |    |-- CC-PVTZ-3P
|    |    |    |-- MOLCAS.template
|    |    |    |-- OPENCAPMD.gbs
|    |    |    |-- MOLCAS.resources
|    |    |    |-- runQM.sh
```

Figure 2: Directory structure of OpenMolcas based calculation.

- `geom`: geometry file in SHARC or Columbus format. The columns in the following correspond to atom symbol, atomic number, x, y, z coordinates in bohr and lastly the atomic masses.

```
C   6.0          -0.033968      -0.214849      -0.021765       12.00000000
H   1.0          -0.093819       0.027585       1.992555       1.00782504
H   1.0           1.770523       0.027597      -0.918910       1.00782504
C   6.0          -2.094619      -0.788029      -1.341295       12.00000000
H   1.0          -2.034768      -1.030463      -3.355614       1.00782504
H   1.0          -3.899110      -1.030473      -0.444149       1.00782504
```

Figure 3: Geometry file structure for `geom` input.

- **QM/**: A directory containing input files for `SHARC_MOLCAS_OPENCAP.py` interface (the file structure is similar to that of `SHARC_MOLCAS.py` interface.).
  - **CC-PVTZ-3P**: Basis set file used for calculations only if a modified basis is used which is not a standard choice from molcas basis library. The basis file should have molcas basis-library file structure. See the file `examples/openmolcas/QM/CC-PVTZ-3P` for reference. This file should have a header as following

    ```
    #Hamiltonian NRH
    #Nucleus UNK
    #Contraction CCC
    #AllElectron AE_
    ```

    and individual atom specific header should be renamed by the basis file name with proper count of uncontracted and contracted basis count in the header.
    (see header in Fig. 4 `/C.cc-pVTZ-3P.Dunning.10s8p2d1f.4s6p2d1f.`).

```
/C.cc-pVTZ-3P.Dunning.10s8p2d1f.4s6p2d1f.
T.H. Dunning. J. Chem. Phys. 90 (1989) 1007-1023. doi:10.1063/1.456153
CARBON (10s,8p,2d,1f) -> [4s,6p,2d,1f]
6.0   3
* s-type functions
10    4
8236.00000000
1235.00000000
280.80000000
79.27000000
25.59000000
8.99700000
3.31900000
0.90590000
0.36430000
0.12850000
0.0005310000 -0.0001130000  0.0000000000  0.0000000000
0.0041080000 -0.0008780000  0.0000000000  0.0000000000

...
```

Figure 4: Custom basis file for `SHARC_MOLCAS_OPENCAP.py`

– **MOLCAS.template**: Template file for `SHARC_MOLCAS_OPENCAP.py`. This is also a file standard to SHARC. This file contains keyword to run the `SHARC_MOLCAS_OPENCAP.py` interface. A sample file with explanation of keywords are given in the following section (Fig. 5). Keywords are not case sensitive.

```
basis cc-pvtz-3p
# Basis set chosen (in openmolcas-library format)
# In this example this is not a standard basisset
# available in molcas library

BASLIB /usr/ethene/SA15-3e20o-CCT3PA/QM/
# Basis file location for openmolcas input file
# if default basisset is not used
# In this example MOLCAS will seek for CC-PVTZ-3P file in BASLIB.
# This modified basisfile name is case sensitive.

ras2 20
# CAS orbital size

nactel 3
# No. of active electrons

inactive 7
# Inactive orbitals

roots 15
# Total no. of states in state averaging

charge -1
# Charge of the system

CAP_ETA 1771.25
# Fixed eta value
# (This value is scaled by 1E-5 in actual calculation.)

act_states 3
# Active state to follow
# (resoance state from initial calculation.)

cap_x 7.62
cap_y 4.20
cap_z 6.02
# CAP onset along all directions in atomic units.

calc_neutral
# Calculate neutral energy
# (if absent, take lowest root as a dummy neutral energy)

track_all
# Keyword for wf overlap
# Use RASSI module of openmolcas to calculate
# Other option is track_wfoverlap (the default choice).

natorb
# Print natural orbitals (real part of complex density)
# Need OPENCAPMD.gbs basis file in pyscf format
# Should be in QM/ directory
```

Figure 5: `MOLCAS.template` file with necessary keywords.

– **OPENCAPMD.gbs**: Gaussian basis set file written in pyscf basis input format (needed if `natorb` is invoked in `MOLCAS.template` file.) The file may look like the Fig. 6.

```
spherical


****
H     0
S   4   1.00
1.301000D+01           1.968500D-02
1.962000D+00           1.379770D-01
4.446000D-01           4.781480D-01
1.220000D-01           5.012400D-01
S   1   1.00
1.220000D-01           1.000000D+00
P   1   1.00
7.270000D-01           1.0000000
****
C     0
S   9   1.00
6.665000D+03           6.920000D-04
1.000000D+03           5.329000D-03
2.280000D+02           2.707700D-02
6.471000D+01           1.017180D-01
2.106000D+01           2.747400D-01
7.495000D+00           4.485640D-01
2.797000D+00           2.850740D-01
5.215000D-01           1.520400D-02
1.596000D-01          -3.191000D-03


....
****
```

Figure 6: A sample file snapshot of `OPENCAPMD.gbs` file read by pyscf.

– **MOLCAS.resources**: Resource file for OpenMolcas jobs. This is also a standard file in SHARC. A sample file looks like the following. Keywords are not case sensitive.

```
memory 8800
# Memory per core (in MB), (not TOTAL Memory)
# If 256 GB Memory is distributed over 16 core,
# this means 16000 is the value to be used.

molcas $MOLCAS
# MOLCAS variable, can also use absolute path as well.

scratchdir /scratch/usr/ethene/3pA/
# Scratch directory

ncpu 1
# No. of cpu for EACH parallel run (NOT total cpus)

driver /usr/molcas_build/pymolcas
# pymolcas module path

WFOVERLAP /usr/WfOverlap/bin/wfoverlap.x
# WF-OVERLAP directory with the binary file (needed)

ncpu_avail 28
# Total no. of cpu for the job

delay 0.01
# Delay between parallel jobs (not needed strictly)

rasorb
# Read casscf orbitals (MOLCAS.RasOrb) files as guess
# for consecutive steps in geometry optimization cycles.
# (other option is scforb for SCF orbital guesses.)

optimizer geometric
# Optimizer chosen (Default is geometric)
```

Figure 7: `MOLCAS.resources` file with necessary keywords.

– **runQM.sh**: Script to execute `SHARC_MOLCAS_OPENCAP.py` interface (default to SHARC). If not present, then it will be created by `optimizer_MOLCAS.py`. A sample file looks like following (similar to SHARC-QC software interfaces).

```
#!/bin/sh
export SPATH=/opencapmd/sharc-interface/SHARC_MOLCAS_OPENCAP.py
cd QM || exit
"$SPATH" QM.in >> QM.log 2>> QM.err
exit $?
```

– **run-opencapmd.sh**: Bash script to execute OpenCAP-MD with OpenMolcas. The optimizer is called via the following command.

```
# Change to the path of the opencap-md
export OPTIMIZER_PATH=/path_TO_opencapmd/optimizer/
export OPTIMIZER="$OPTIMIZER_PATH/optimizer_MOLCAS.py"
$OPTIMIZER `pwd` > logfile
```

`optimizer_MOLCAS.py` takes command line argument of the directory path where the job is supposed to run.

- **Outputs created**: The output files created during geometry optimizations.

- **iteration_details**: Logs iteration details from the optimization process.
- **geomeTRIC.log, opt_log.out, berny.log**: A logfile generated by the `geometric`, `optking`, `pyberny` optimizers respectively.
- **QM/QM.log**: Log file from `SHARC_MOLCAS_OPENCAP.py` execution.
- **QM/PCAP.out**: Complex and neutral reference energies generated during the optimization.

2.2 Additional keywords

Keyword	Description
`no_capgrad_correct`	If user wants to skip the CAP gradient calculation.
`ghost`	Add ghost atom in center of mass (COM). This demands presence of a basis set in molcas format in `QM/X.basis file`.
`scforb`	If scf orbitals are used as guess for next iteration.
`track_wfoverlap`	If WF-OVERLAP package is used to calculate wave-function overlaps.
`screening`	Keyword followed by a float value (say, 0.0001) which works as a thershold to remove gradient and coupling jobs that have insignificant contributions.

Table 1: Description of some additional `MOLCAS.template` keywords

Keyword	Description
`scforb`	If SCF orbitals are used as guess for next iteration.
`always_guess`	If at each step new SCF guess is used.
`debug`	If all electronic structure calculation files are stored for debugging purpose (requires significant disk-space).
`optimizer`	three options: geometric, optking, pyberny

Table 2: Description of some additional `MOLCAS.resources` keywords

2.3 Special constraints/parameters for optimizers

2.3.1 geometric, pyberny

The keywords in this following section are optimizer specific and can be added to `MOLCAS.resources` file and the optimizer will adapt accordingly.

Threshold keyword	Description
`gradientmax`	Maximum gradient in a.u..
`gradientrms`	RMS gradient in a.u..
`stepmax`	Maximum displacement in a.u..
`steprms`	RMS displacement in a.u..
`deltae`	Energy convergence in a.u..

Table 3: Description of optimizer specific `MOLCAS.resources` threshold keywords

2.3.2 optking

Create a file `optking.params` in run directory where `optking` optimizer specific keywords are invoked (see Psi4 OptKing Module Documentation). A sample input may look like Fig. 8

```
g_convergence qchem
ADD_AUXILIARY_BONDS True
opt_coordinates both
INCLUDE_OOFP False
```

Figure 8: A sample file for `optking.params`.

These above keywords are specific for `optking` module. There are several ways the same convergence parameters shown in Table 3 can be invoked for `optking` by changing the `g_convergence` keyword in `optking.params`. A detailed logistics and available options are in optking convergence algorithm document.

2.4 Columbus specific files

For reference please look at the file structure needed in the `examples/columbus` directory as shown in Fig. 9.

```
OPENCAP-MD
|-- examples
|   |-- columbus
|   |   |-- COLUMBUS.template
|   |   |-- COLUMBUS.resources
|   |   |-- run-opencapmd.sh
|   |   |-- geom
|   |   |-- optking.params
|   |   |-- colfiles/
```

Figure 9: Directory structure of Columbus based calculation.

- **COLUMBUS.template**: Input template for Columbus based optimization.

```
nstates 5
# Total no of states in MR-CIS/MCSCF calculation calculation

eta_opt 0.00848
# optimized eta chosen (Scaled by 1.0)
# (In atomic units.)

act_state 2
# Active state to track

cap_x 2.77
cap_y 2.77
cap_z 4.88
# cap onset in x, y and z directions
# in atomic units

natorb
# Print natural orbitals in DIAG basis
# (OPENCAPMD.gbs basis file is needed in pyscf format)

maxsqnorm 0.998001
# Maximum square norm as a cut-off.
# c^2 sum limits printing of all CSFs and Slater determinants
# (Applicable for WF-OVERLAP calculation only).

symmetry
# If point group Symmetry is invoked.

ghost_atom
# If ghost atom should be added at COM
```

2.5 Additional keywords

Keyword	Description
`calculation`	Followed by a string. For SA-CASSCF calculation use `mcscf`. Default is MR-CIS, hence this keyword is not needed.
`track`	Followed by a string. Tracking method during geometry optimization. (When symmetry is on) two options are there: `civecs` and `dens`, which are done by CI-vector overlap and attachment-detachment densities, respectively. Default is `dens`. (When symmetry is off (C_1)) WF-OVERLAP is the default choice and this keyword is redundant.
`no_capgrad_correct`	If user wants to skip the CAP gradient calculation.
`screening`	Keyword followed by a float value (say, 0.0001) which works as a thershold used to remove gradient and coupling jobs that have insignificant contributions.
`ghost_atom`	Add ghost atom in center of mass (COM).

Table 4: Description of some additional `COLUMBUS.template` keywords

- **COLUMBUS.resourcses**: Input resource file for Columbus simulations. The keywords are not case sensitive.

```
WFOVERLAP/usr/wf-overlap/bin/wfoverlap.x
# wf-overlap binary file location

memory 240000
# Total Memory available

ncpu  16
# Total cpu available

optimizer optking
# Optimizer chosen optking (if Symmetry is on)

inputdir /usr/OPENCAPMD/N2/MRCIS/colfiles
# Input files for COLUMBUS run
# (If inputdir key is absent, default is 'copyfiles/' directory
# in the working directory.)
```

Keyword	Description
`columbus`	Columbus absolute path. If absent, the script will try to read `$COLUMBUS` variable from command line.
`scratchdir`	Scratch directory where parallel gradient and coupling calculations are performed, this is disk-space intensive. If absent, the path is `SAVEDIR/scratch` in current working directory with same file-tree structure.
`savedir`	Absolute path where all Columbus electronic structure package runs it calculation. If absent, `SAVEDIR` is prepared in current work directory. Within this directory, folders from each iteration is present: `sp_0`, `sp_1`, `sp_2`, `sp_3` and so on. Each folder will contain all Columbus run files and needed for debugging and `restart`.
`restart`	Followed by an integer value, that is the iteration number. This demands that `savedir` is accessible. Otherwise the restart will fail. The user is asked to resubmit a new job with the last geometry in a new directory.
`optimizer`	Optimizers available. Default is `optking`. Additionally you can use `optking.params` for more parameter control of the `optimizer`. (When symmetry is on) Use optking. (When symmetry is off (C_1)) Use geometric
`debug`	All Columbus single point calculation directories are saved (with Columbus `WORK` directory). Disk-space intensive.

Table 5: Description of some additional `COLUMBUS.resources` keywords

- **run-opencapmd.sh**: Execution script for Columbus calculations. Same to that of OpenMolcas specific optimizer submission.

```
# Change to the path of the opencap-md
export OPTIMIZER_PATH=/path_TO_opencapmd/optimizer/
export OPTIMIZER="$OPTIMIZER_PATH/optimizer_columbus.py"
$OPTIMIZER `pwd` > logfile
```

- **optking.params**: Configuration file for optimization settings specific to `optking` optimizer. See section2.3.2.

```
colfiles/
|-- mcdrtin.*, mcscfin
# (and all MCSCF related files)
|-- ANION/
|--|-- ciudgin
# (MR-CI related files or MC-SCF restart files)
|--|-- ABACUSIN, ABACUSIN.NAD (# and all other files)
# (These gradient and NAC specific files should be present)
|-- NEUTRAL/
```

Figure 10: Directory structure of Columbus files specific for projected-CAP calculation

- **colfiles/**: Contains all Columbus-related input `$COLUMBUS/colinp` generated output files (not expanded here for brevity, see `examples/columbus/colfiles` for reference). User has to create the reference files that will be used for different iterations. These absolute path containing these files should be given in the `inputdir` keyword in the `COLUMBUS.resources` file. The main structure of the columbus is shown in 2.5 file tree.

  The `inputdir` should have the MCSCF files generated by the user using `$COLUMBUS/colinp`. For reference, the Columbus specific `control.run` file looks like the following.

```
niter = 1
hermit
scf
mcscf
```

  Within the main `inputdir`, there are two main directories: `ANION` and `NEUTRAL`.

  The `ANION` folder should have files to calculate MR-CIS energies, transition moments, gradients and couplings. The set up of `ANION` folder should have `ABACUSIN.NAD` and `ABACUSIN` for couplings and gradients respectively. Even though `slope` and `nadcoupl` is present in `control.run` file, these will be calculated in parallel after `pciudg` keyword is executed. The initial MO for this calculation is from `mcscf` run in the previous directory. For reference, the Columbus specific `control.run` file looks like the following.

```
niter = 1
hermit
ciudgmom
mcscf
pciudg
slope
nadcoupl
```

  The `NEUTRAL` folder should have files for MR-CI calculation for neutral state. The initial MO for this calculation is from `mcscf` run in the previous directory.

```
niter = 1
hermit
ciudg
```

2.6 Working principle of `optimizer_columbus.py`

This module is heavily dependent on *apriori* generation of Columbus iinput files using `$COLUBUS/colinp`. Given that the `colfiles/` are present in the work directory.

- Changes `geom` file in each iteration (run in `savedir/sp_0` for iteration 0)
- Executes `$COLUMBUS/geom.unik.x` to get symmetry unique atoms and coordinates in `geom.unik` file.
- Changes the `daltaoin` file with new symmetry unique atoms and coordinates. (Needed for `$COLUMBUS/dalton.x` for AO integral generation)
- Perform SA-CASSCF calculation in the `savedir/sp_0` (say, for iteration 0) directory. Copy the MO guess (`mocoef`) in `ANION`, `NEUTRAL` folder.
- Run the Columbus job in `ANION` folder and generate the energies and transition moments for anionic states. Make sure that you use the parallelized module `$COLUMBUS/pciudg.x` instead of `$COLUMBUS/ciudg.x` to save a lot of computation time for MR-CI calculations.

  After the energies and 1-RDMs are calculated, gradients and couplings are calculated in parallel.
- Once `ANION` folder execution is done, switch to `NEUTRAL` folder and calculate neutral energy.
- Finally, when all jobs are successful Projected CAP files needed for `OpenCAP` is generated and `OPENCAP-MD` is called to calculate complex energies and CAP-gradient augmented gradients.
- Iteration specific energy and gradients are fed to the `optimizer`.