MOMAP

Tutorial 07

Dalton Calculation

Version 2020A

May, 2020

MOMAP Tutorial 07

Version 2020A edited by:

Dr. Qikai Li

Dr. Deping Hu

Released by Hongzhiwei Technology (Shanghai) Co., Ltd

and Z.G. Shuai Group

The information in this document applies to version 2020A of MOMAP

TABLE OF CONTENTS

MOMAP Dalton Tutorial

BACKGROUND

The Spin-Orbit (SO) interaction is a well-known phenomenon that manifests itself in lifting the degeneracy of one-electron energy levels in atoms, molecules, and solids. In solid-state physics, the nonrelativistic Schrodinger equation is frequently used as a first approximation, e.g. in electron band-structure calculations. Without relativistic corrections, it leads to doubly-degenerated bands, spin-up and spin-down, which can be split by a spin-dependent term in the Hamiltonian. In this approach, spin-orbit interaction can be included as a relativistic correction to the Schrodinger equation.

The SO interaction effect is always present, and gives corrections to the total energy and its derivatives. Actually, the strength of the SO coupling increases quickly with the atomic number Z: as inner-shell electrons are pulled closer to the nucleus, their kinetic energy increases and relativistic effects become very important. In many cases, for light elements, these can be neglected, or approximated by the scalar relativistic terms in the Dirac equation. However, for specific properties, SO effects might be important even when only light elements are present, as for graphite. In second-row transition metals and heavier elements, but also for some lighter elements, the SO effect is essential to reproduce correctly the electronic structure of materials. Classic examples include the valence band splitting of GaAs, and the multiplet structure of the f-band metals. For heavier elements, in general, the SO effect becomes as important for structural and dynamical properties as for electronic properties. In addition, for bulk structures SO should be used if heavier elements are included (late d-metals, f-metals), and for surfaces SO should be considered as well due to anisotropy of interface with vacuum.

MOMAP PySOC can be used to calculate the spin-orbit coupling (SOC) elements between singlet and triplet states, including both ground and excited states. The SOC plays a fundamental role in spin-forbidden excited-state processes, such as intersystem crossing and phosphorescence. From the computational chemistry standpoint, with the increasing popularization of dynamics simulations for studying excited states and charge transport, there is an increasing demand for new methods to efficiently evaluate SOC elements. PySOC targets this demand, with SOC computations using DFT-based methods. In the current version, PySOC is interfaced to Gaussian g09/g16 and DFTB+ codes, while the atomic integrals in PySOC are calculated by the MolSOC code developed by Sandro Giuseppe Chiodo *et al*. SOCs are evaluated on the basis of time-dependent density functional theory (TDDFT), TDDFT with Tamm-Dancoff approximation (TDA), and time-dependent density

functional tight binding (TD-DFTB); all three solved within linear-response approximation and using Casida's wave functions. Calculations with PySOC are very fast. The initial linear-response calculation is typically the computational bottleneck of SOC evaluations, and the final cost is basically that of computing energies for the singlet and triplet states of interest.

An alternative way to calculate the SOC at the TDDFT level is to use the DALTON[7] program, which is free and open-source, and has been integrated with MOMAP. In doing the SOC calculation, the scaled spin-orbit integrals and the atomic-mean-field approximation can be used and the relativistic effects can be included using Douglas-Kroll-Hess second-order one-electron scalar integrals. The DALTON can also be used to calculate the "transition moment" of triplet states with quadratic response theory, which is very useful in the prediction of phosphorescence rate.

References:

1. E. K. U. Gross and R. M. Dreizler, LDA Density Approximations in Quantum Chemistry and Solid State Physics (Plenum, 1986), pp. 353–379.
2. C. L. Kane and E. J. Mele, Phys. Rev. Lett., 2005, 95, 226801.
3. M. P. Surh, M.-F. Li, and S. G. Louie, Phys. Rev. B, 1991, 43, 4286.
4. M. Divis, M. Richter, H. Eschrig, and L. Steinbeck, Phys. Rev. B, 1996 53, 9658.
5. X. Gao, S. Bai, D. Fazzi, T. Niehaus, M. Barbatti, and W. Thiel, Evaluation of spin-orbit couplings with linear-response time-dependent density functional methods, J. Chem. Theory Comput., 2017, 13, pp 515-524.
6. Sandro Giuseppe Chiodo, Monica Leopoldini, MolSOC: A spin-orbit coupling code, Computer Physics Communications, 2014, 185, pp 676-683.
7. DALTON website: https://www.daltonprogram.org/ (https://doi.org/10.1002/wcms.1172)

QUICK START

Once MOMAP is properly installed, the Dalton package is located under **$MOMAP_ROOT/dalton** directory, and looks like the following:

```
[MOMAP-2020A]$ tree dalton
dalton
├── GIT_HASH
├── VERSION
├── basis
│   ├── 3-21++G
│   ├── 3-21++G*
│   ├── 3-21G
│   ├── 3-21G*
│   ├── 4-31G
│   ├── 6-31++G
│   ├── 6-31++G*
│   ├── 6-31++G**
│   ├── 6-31+G
│   ├── 6-31+G*
│   ├── 6-311++G(2d,2p)
│   ├── 6-311++G(3df,3pd)
│   ├── 6-311++G**
│   ├── 6-311+G*
│   ├── 6-311G
│   ├── 6-311G(2df,2pd)
│   ├── 6-311G*
│   ├── 6-311G**
│   ├── 6-31G
│   ├── 6-31G(3df,3pd)
│   ├── 6-31G*
│   └── 6-31G**
…
├── dalton
├── dalton.x
├── dalton_get_info.py
├── dalton_prepare.py
└── tools
…

10 directories, 390 files
```

To facilitate Dalton calculation with MOMAP, we have added several handy python scripts to prepare for the Dalton jobs:

- `dalton_gjf2mol.py`
- `dalton_xyz2mol.py`
- `dalton_prepare.py`
- `dalton_get_info.py`

Users can use the first two python scripts to prepare the Dalton mol file. Use `-h` or `--help` to see how to use the python script. For example,

```
[MOMAP]$ ~/MOMAP-2020A/dalton/dalton_get_info.py --help
Usage:
    dalton_get_info.py dalton.out
```

The basic steps involved to do Dalton calculation with MOMAP are as follows:

1. Prepare the Dalton mol input file, for example, by using `dalton_gjf2mol.py` or `dalton_xyz2mol.py`
2. Prepare the MOMAP control file `momap.inp`
3. Use `momap.py` to carry out calculations.

That's it.

PREPARE DALTON MOL FILE

■ From Gaussian gjf / com file

Use `dalton_gjf2mol.py` to do the conversion. We use `benzaldehyde` as an example, a typical `.com` file for benzaldehyde is shown as follows:

```
[dalton]$ cat s1.com
%chk=s1.chk
# opt freq hf/3-21g

s1

0 1
 C              -3.73319912   -0.55712190   -0.16925411
 C              -2.37835267   -0.57337757   -0.14291193
 C              -1.66623688    0.62289202    0.01756275
 C              -2.35515545    1.83597157    0.15079624
 C              -3.71000188    1.85222733    0.12445331
 C              -4.42211769    0.65595762   -0.03602037
 H              -4.27691530   -1.47050010   -0.29178158
 H              -1.85234803   -1.49959061   -0.24463967
 H              -0.59651602    0.61005739    0.03836076
 H              -1.81143921    2.74935011    0.27332084
 H              -4.23600647    2.77844061    0.22617901
 C              -5.82315209    0.67276744   -0.06326038
 O              -6.48440249   -0.43805445   -0.21227156
 H              -6.34915671    1.59898056    0.03846668
```

The converted mol file is shown as follows:

```
[dalton]$ cat s1.mol
ATOMBASIS
Generated by MOMAP
through gjf2mol.py
Atomtypes=3 Charge=0 Angstrom NoSymmetry
Charge=8.0 Atoms=1 Basis=3-21G
O      -6.624639     -0.366782     -0.207243
Charge=6.0 Atoms=7 Basis=3-21G
C      -3.719449     -0.576840     -0.171374
C      -2.333845     -0.567037     -0.141329
C      -1.629876      0.624964      0.018395
C      -2.332137      1.826888      0.150011
C      -3.713231      1.843764      0.123386
C      -4.437544      0.636680     -0.038491
C      -5.855595      0.662291     -0.064834
Charge=1.0 Atoms=6 Basis=3-21G
H      -4.258167     -1.509588     -0.296198
H      -1.794804     -1.503817     -0.244132
H      -0.545581      0.621354      0.040553
H      -1.790669      2.759467      0.274745
H      -4.250471      2.781998      0.226379
H      -6.407538      1.605826      0.038065
```

■ From XMOL xyz file

a) Prepare the dalton mol file from `xyz` file as follows:

```
$ dalton_xyz2mol.py s1.xyz -b 3-21G
```

Note: Use option `-b` or `--basis` to set the `basis`.

```
[dalton]$ cat s1.xyz
  14
scf done:  -345.449312
 C    -3.719449    -0.576840    -0.171374
 C    -2.333845    -0.567037    -0.141329
 C    -1.629876     0.624964     0.018395
 C    -2.332137     1.826888     0.150011
 C    -3.713231     1.843764     0.123386
 C    -4.437544     0.636680    -0.038491
 H    -4.258167    -1.509588    -0.296198
 H    -1.794804    -1.503817    -0.244132
 H    -0.545581     0.621354     0.040553
 H    -1.790669     2.759467     0.274745
 H    -4.250471     2.781998     0.226379
 C    -5.855595     0.662291    -0.064834
 O    -6.624639    -0.366782    -0.207243
 H    -6.407538     1.605826     0.038065
```

The XMOL xyz file format is very simple, and is represented by a two-line "header", followed by one line for each atom. The converted mol file is shown as follows:

```
[dalton]$ cat s1.mol
ATOMBASIS
Generated by MOMAP
through xyz2mol.py
Atomtypes=3 Charge=0 Angstrom NoSymmetry
Charge=8.0 Atoms=1 Basis=3-21G
O      -6.624639     -0.366782     -0.207243
Charge=6.0 Atoms=7 Basis=3-21G
C      -3.719449     -0.576840     -0.171374
C      -2.333845     -0.567037     -0.141329
C      -1.629876      0.624964      0.018395
C      -2.332137      1.826888      0.150011
C      -3.713231      1.843764      0.123386
C      -4.437544      0.636680     -0.038491
C      -5.855595      0.662291     -0.064834
Charge=1.0 Atoms=6 Basis=3-21G
H      -4.258167     -1.509588     -0.296198
H      -1.794804     -1.503817     -0.244132
H      -0.545581      0.621354      0.040553
H      -1.790669      2.759467      0.274745
H      -4.250471      2.781998      0.226379
H      -6.407538      1.605826      0.038065
```

Please check the generated mol file for any abnormalities. From the generated mol file, we can see that all the basic fields are filled and it seems to be fine.

CREATE CONTROL FILE MOMAP.INP

The MOMAP `momap.inp` for doing Dalton calculation is straightforward, as shown below:

```
[dalton]$ cat momap.inp
do_dalton      = 1

&dalton
  sched_type   = pbs                ! default to 'pbs', can be pbs, slurm, lsf or local
  qc_queue     = X12C               ! default to 'workq'
  # module_qc  = dalton/2018.0      ! default to ''

  qc_exe = dalton                   ! default to 'dalton'
  qc_ppn = 8                        ! default to 1

  t1_s0_soc     = 1                 ! default to 0
  tn_sn_soc     = 1                 ! default to 0
  t1_trans_dip = 1                  ! default to 0

  ex_molfile    = s1.mol

  functional    = 'B3LYP'           ! default to 'B3LYP'
  nstates = 3                       ! default to 3
/
```

Note:

Most parameters are optional, and having their defaults, users only need to set the key parameters for the specific case.

USE MOMAP.PY TO DO CALCULATION

In a typical Dalton calculation, one needs only a Dalton geometry input file `*.mol`, a MOMAP control file `momap.inp`, and optionally a run script file `run.sh`. Once the files are ready, we can use `momap.py` to do the calculation by using the following command:

```
[dalton]$ ./run.sh
```

The `run.sh` script is shown as follows:

```
[dalton]$ cat run.sh
#!/bin/sh
python $MOMAP_ROOT/bin/momap.py  1> log 2>&1 &
```

The "`1> log 2>&1`" means to join the `stdout` and `stderr`, and redirect to file `log`.

The `momap.py` will first call `dalton_prepare.py` to generate the Dalton `*.dal` files and a job submission script `run_job.*`, based on the `momap.inp` and environmental settings.

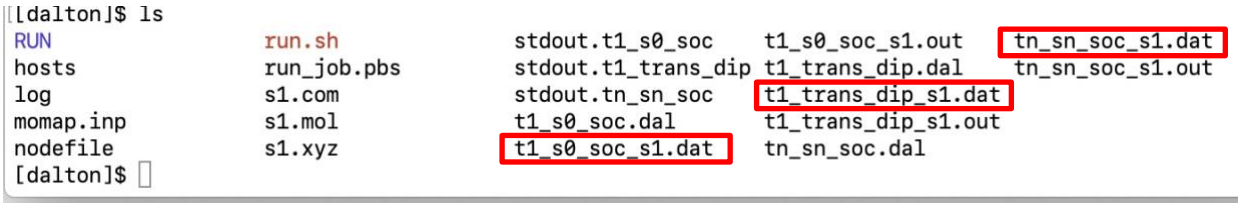
A typical job script file is as shown as follows:

```
[dalton]$ cat run_job.pbs
#PBS -l walltime=1000:00:00
#PBS -l nodes=1:ppn=8
#PBS -q X12C
#PBS -o stdout
#PBS -j oe

if test -f 'hosts'; then
  rm -f hosts
fi
echo ${PBS_NODEFILE} > hosts

cd ${PBS_O_WORKDIR}

$MOMAP_ROOT/dalton/dalton -mb 120 -noarch -nodelist hosts -N 8 t1_s0_soc
s1 &> stdout.t1_s0_soc

rm -f RUN/running.dalton-t1_s0_soc

$MOMAP_ROOT/dalton/dalton -mb 120 -noarch -nodelist hosts -N 8 tn_sn_soc
s1 &> stdout.tn_sn_soc

rm -f RUN/running.dalton-tn_sn_soc

$MOMAP_ROOT/dalton/dalton -mb 120 -noarch -nodelist hosts -N 8
t1_trans_dip s1 &> stdout.t1_trans_dip

rm -f RUN/running.dalton-t1_trans_dip
```

In the above-mentioned scripts, we will carry out calculations on:

- `t1_s0_soc`
- `tn_sn_soc`
- `t1_trans_dip`

When the calculations are finished, the final file layout is shown as follows:



All the treated Dalton output results are in `*.dat` files as shown above:

```
[dalton]$ cat t1_s0_soc_s1.dat
Parsing Spin-orbit coupling constants...
 Hso_x:       47.978703 cm-1
 Hso_y:       56.403667 cm-1
 Hso_z:        7.920588 cm-1
 Hso_ave:     42.996377 cm-1
Parsing polarization partial rates...
Parsing transition dipole...
```

```
[dalton]$ cat t1_trans_dip_s1.dat
Parsing Spin-orbit coupling constants...
Parsing polarization partial rates...
 dipole_sub_state_x:    3.851000e-03 au
 dipole_sub_state_y:    8.813000e-04 au
 dipole_sub_state_z:    3.453000e-04 au
 dipole_average(mod):   1.007941e-02 debye
Parsing transition dipole...
 dipole_sub_state_x:    3.851000e-03 au
 dipole_sub_state_y:    8.813000e-04 au
 dipole_sub_state_z:    3.453000e-04 au
 dipole_average(mod):   1.007941e-02 debye
```

```
[dalton]$ cat tn_sn_soc_s1.dat
Parsing Spin-orbit coupling constants...
Parsing polarization partial rates...
Parsing transition dipole...
-----------------------------------------------
    B         C        Hso, cm^-1
-----------------------------------------------
  1  1  0    1  1  1      -0.006584
  1  1  0    1  1  1       0.006584
  1  1  0    1  1  1      -0.186553
  2  1  0    1  1  1      -8.917254
  2  1  0    1  1  1     -11.460965
  2  1  0    1  1  1      -1.591191
  3  1  0    1  1  1     -26.251361
  3  1  0    1  1  1     -30.269941
  3  1  0    1  1  1      -4.277561
  1  1  0    2  1  1     -24.818191
  1  1  0    2  1  1     -29.231826
  1  1  0    2  1  1      -4.117344
  2  1  0    2  1  1       0.002195
  2  1  0    2  1  1       0.002195
  2  1  0    2  1  1      -0.006584
  3  1  0    2  1  1       0.184359
  3  1  0    2  1  1       0.478455
  3  1  0    2  1  1       0.041700
  1  1  0    3  1  1      -0.805472
  1  1  0    3  1  1      -1.733850
  1  1  0    3  1  1      -0.230448
  2  1  0    3  1  1       0.000000
  2  1  0    3  1  1      -0.006584
  2  1  0    3  1  1       0.002195
  3  1  0    3  1  1       0.002195
  3  1  0    3  1  1       0.004389
  3  1  0    3  1  1       0.006584
-----------------------------------------------
Note: The first 6 columns correspond to the
     excited state no., symmetry, spin of B, &
     excited state no., symmetry, spin of C
```

The job is done.

INTEL IMPI ENVIRONMENT

To run the Dalton job in parallel, one may need to use the intel IMPI environment. First download the intel IMPI package from the link: http://www.momap.net.cn/index.php/downloads/ under the section "Related software". Move the downloaded file to the desired installation directory and unzip the file by command:

```
$ tar xzf IMPI-2019.5.281-linux-x86_64.tar.gz
```

Then add the following line to the job submission script file:

```
source <installed_dir>/IMPI/intel64/bin/mpivars.sh
```

Or modify the `MOMAP_ROOT/dalton/dalton_prepare.py` once and for all.

The job submission script file may look like the following:

```
[dalton]$ cat run_job.pbs
#PBS -l walltime=1000:00:00
#PBS -l nodes=1:ppn=8
#PBS -q X12C
#PBS -o stdout
#PBS -j oe

if test -f 'hosts'; then
  rm -f hosts
fi
echo ${PBS_NODEFILE} > hosts

cd ${PBS_O_WORKDIR}

source <installed_dir>/IMPI/intel64/bin/mpivars.sh

$MOMAP_ROOT/dalton/dalton -mb 120 -noarch -nodelist hosts -N 8 t1_s0_soc
s1 &> stdout.t1_s0_soc

rm -f RUN/running.dalton-t1_s0_soc

$MOMAP_ROOT/dalton/dalton -mb 120 -noarch -nodelist hosts -N 8 tn_sn_soc
s1 &> stdout.tn_sn_soc

rm -f RUN/running.dalton-tn_sn_soc

$MOMAP_ROOT/dalton/dalton -mb 120 -noarch -nodelist hosts -N 8
t1_trans_dip s1 &> stdout.t1_trans_dip
```

If the environment module is used in the computing cluster, one may not need to modify the job submission script, suppose the intel IMPI module name is `intel/impi/2019.5.281`, we can simply modify the `momap.inp` as follows:

```
[dalton]$ cat momap.inp
do_dalton     = 1

&dalton
  sched_type   = pbs                ! default to 'pbs', can be pbs, slurm, lsf or local
  qc_queue     = X12C
  module_qc    = dalton/2018.0 intel/impi/2019.5.281

  qc_exe = dalton                   ! default to 'dalton'
  qc_ppn = 8                        ! default to 1

  t1_s0_soc     = 1                 ! default to 0
  tn_sn_soc     = 1                 ! default to 0
  t1_trans_dip = 1                  ! default to 0

  ex_molfile    = s1.mol

  functional    = 'B3LYP'           ! default to 'B3LYP'
  nstates = 3                       ! default to 3
/
```

Then, we can do the Dalton calculations as usual.