MOMAP

Tutorial 06

PySOC Calculation

Version 2020A

April, 2020

MOMAP Tutorial 06

Version 2020A edited by:

Dr. Qikai Li

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 PySOC 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.

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.

Quick Start

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

```
[MOMAP]$ tree pysoc
pysoc
├── bin
│   ├── dftb+.exe
│   ├── dp_bands.exe
│   ├── dp_dos.exe
│   ├── gen2cif.exe
│   ├── gen2xyz.exe
│   ├── makecube.exe
│   ├── modes.exe
│   ├── molsoc0.1.exe
│   ├── repeatgen.exe
│   ├── soc.py
│   ├── soc_prepare.py
│   ├── soc_td.exe
│   ├── straingen.exe
│   ├── waveplot.exe
│   └── xyz2gen.exe
├── examples
│   ├── ch2o_gaussian
│   │   ├── AO_overlap
│   │   ├── MOA_coeffs
…
├── input_template
│   └── init.py
├── lib
│   └── python2.7
│       └── site-packages
│           ├── dptools
│           │   ├── __init__.py
…
└── parameters
    └── mio-1-1
    │   ├── C-C.skf
    │   ├── C-H.skf
    │   ├── C-N.skf
    │   ├── C-O.skf
    │   ├── C-P.skf
    │   ├── C-S.skf
…
    └── mio-1-1-fit
        ├── C.basis
        ├── H.basis
        ├── N.basis
        ├── O.basis
        ├── P.basis
        └── S.basis

13 directories, 158 files
```

The basic steps involved to do PySOC calculation are as follows:

1. Prepare QM calculation input file
2. Prepare `momap.inp`
3. Use `momap.py` to carry out calculations.

That's it. Now let us dig a little into the details.

Excited States Electronic Structure QM Calculation

■ Gaussian 09/16

Prepare the Gaussian input `com` file with the following suggested must-have settings:

```
%rwf=gaussian.rwf
# td(50-50,nstates=5)wB97XD/TZVP 6D 10F nosymm GFInput
```

The keywords, `50-50`, `nstates=5`, mean 5 singlets and 5 triplets will be calculated.

Note:

a) The `gaussian.rwf`, `6D`, `10F`, `GFInput` key words are necessary, and the string `gaussian` is to be replaced with your own molecular name, for example, `ch2o0`. The `com`, `log`, `chk`, and `rwf` file name stub should be the same.

b) When setting the basis, please check the max layer for each element which should be less or equal than `f` shell. (The higher level like `g` shell is not available in the following SOC calculation at the moment.)

A typical `.com` file is shown as follows:

```
[ch2o_gaussian]$ cat ch2o0.com
%mem=1GB
%nprocs=8
%chk=ch2o0.chk
%rwf=ch2o0.rwf
# td(50-50,nstates=5) wB97XD/TZVP 6D 10F nosymm GFInput

test

0 1
C        -0.131829     -0.000001     -0.000286
O         1.065288      0.000001      0.000090
H        -0.718439      0.939705      0.000097
H        -0.718441     -0.939705      0.000136
```

■ TD-DFTB+

(Strongly suggested: Read the manual before using it)

a) Prepare geometry file from `*.xyz` file:

```
xyz2gen.exe *.xyz to *.gen
```

Note:

The `xyz2gen.exe` is a kind of geometry generation tool from `dptools` for `td-dftb+` and should be installed, and work properly.

b) Prepare `dftb_in.hsd` for `td-dftb+` input:

Besides the general settings, the following key words should be added in the `.hsd` input.

Set parameters: HubbardDerivatives for related elements.

```
set WriteTransitions = Yes
set WriteTransitionDipole = Yes
set WriteEigenvectors = Yes
```

```
set WriteXplusY = Yes
set WriteHS = No
```

c) run `tddftb+` calculation

d) run `tddftb+` once again (this step should be very fast, as it just read the parameterized matrix elements) in the same directory, but with the following changes:

```
set WriteHS = Yes
```

A typical `dftb_in.hsd` file is shown as follows:

```
[ch2o_tddftb]$ cat dftb_in.hsd
Geometry = GenFormat {
  <<< "ch2o.gen"
}

Driver = {}
…
```

Note:

The QM calculation by using Gaussian g09/g16 are recommended, as the TD-DFTB+ needs to be fully tested.

Create Control File momap.inp

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

For Gaussian `g09/g16`

```
[ch2o_gaussian]$ cat momap.inp
do_pysoc      = 1

&pysoc
  sched_type   = local                ! can be pbs, slurm, lsf, or local
  qc_queue     = X12C

  qc_exe       = g09                  ! g09 or g16
  qc_ppn       = 8
  module_qc    = gaussian/g09.e01     ! optional

  pysoc_QM_code = 'gauss_tddft'       ! gauss_tddft or tddftb
  pysoc_QM_input_file = ch2o0.com     ! used only by gauss_tddft

  n_excited_singlets = 4
  n_excited_triplets = 4
/
```

For `TD-DFTB`:

```
 [ch2o_tddftb]$ cat momap.inp
  do_pysoc    = 1

 &pysoc
   sched_type  = local            ! can be pbs, slurm, lsf, or local
   qc_queue    = X12C

   qc_exe = dftb+.exe
   qc_ppn = 8

   pysoc_QM_code = 'tddftb'      ! gauss_tddft or tddftb

   n_excited_singlets = 4
   n_excited_triplets = 4
/
```

Note:

The `qc_ppn` should be compatible with the `nprocs` setting in QM input file. More parameters can be added, please refer to the MOMAP User Guide for details.

Use momap.py to do calculation

In a typical PySOC calculation, one needs only a QM calculation input file `*.com` or `dftb_in.hsd`, 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:

```
[ch2o_gaussian]$ ./run.sh
```

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

```
[ch2o_gaussian]$ cat run.sh
#!/bin/sh

python $MOMAP_ROOT/bin/momap.py -n 8 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 `soc_prepare.exe` to generate two files, that is, `init.py` and a job submission script `run_job.*`, based on the `momap.inp` and the environmental settings.

Users should check the generated `init.py` carefully in the process of running, and stop the job if abnormal settings are found.

A typical job script file is as shown as follows:

```
[ch2o_gaussian]$ cat run_job.local
#!/bin/sh

g09 ch2o0.com

rm -f RUN/running.pysoc
```

A typical input file `init.py` is as shown as follows:

```
[ch2o_gaussian]$ cat init.py
# module called by soc.py
# general control for spin-orbit coupling calculation

import os
import sys

# Control parameters
QM_ex_flag = False          # False we do QM calculation separately
QM_code = 'gauss_tddft'     # gauss_tddft or tddftb
n_s = [1, 2, 3, 4]          # number of excited singlets
n_t = [1, 2, 3, 4]          # number of excited triplets
n_g = ['True']              # default to including ground state
soc_scal = 1                # scaling factor for Zeff in SOC operator
cicoeff_thresh = [1e-05]    # threshhold for ci coeff

# molsoc code from Sandro Giuseppe Chiodo
# with small modifications for input because only the soc
# in atomic basis is needed in the following calculation

MOMAP_ROOT = os.getenv('MOMAP_ROOT')
if not MOMAP_ROOT:
    print ('Please set environment variable MOMAP_ROOT!')
    sys.exit(1)
dir_para_basis = MOMAP_ROOT + '/pysoc/parameters/mio-1-1-fit'

# Input files
if QM_code == 'gauss_tddft':
# from Gaussian output
    qm_out = ['ch2o0.log', 'ch2o0.rwf']
    geom_xyz = []
    soc_key  = ['ANG', 'Zeff', 'DIP']
elif QM_code == 'tddftb':
# from TD-DFTB+ output
    qm_out = ['band.out', 'EXC.DAT', 'oversqr.dat', 'eigenvec.out', 'XplusY.DAT', 'SPX.DAT']
    geom_xyz = ['dty.xyz']
    soc_key  = ['ANG', 'Zeff', 'DIP', 'TDB']

# input for molsoc(to be generated)
molsoc_input = ['molsoc.inp', 'molsoc_basis']
```

Note:

The PySOC needs python 2.7 or above to run!

When calculation is finished, the final file layout is as follows:

```
[ch2o_gaussian]$ ls
RUN             ch2o0.com       ch2o0.rwf       log             pysoc_output.dat
ch2o0.chk       ch2o0.log       data            momap.inp       run.sh
[ch2o_gaussian]$
```

All the calculation related files are moved to directory `data`, and the PySOC results are in file `pysoc_output.dat`.

Check Output Results

Once the run is successful, the SOC elements should be found in output file pysoc_output.dat, and looks like the following:

```
[ch2o_gaussian]$ cat pysoc_output.dat
  __ __    ___    __ __       _       ____
 |  \/  |  / _ \  |  \/  |    / \     |  _ \
 | |\/| | | | | | | |\/| |   / _ \    | |_) |
 | |  | | | |_| | | |  | |  / ___ \   |  __/
 |_|  |_|  \___/  |_|  |_| /_/   \_\ |_|       Version 2020A (2.2.0)

 Copyright (c) 2017 Shuaigroup @ Tsinghua University &
         Institute of Chemistry, Chinese Academy of Sciences.
         All Rights Reserved.

Cite PySOC work as (required):
1. Xing Gao, Shuming Bai, Daniele Fazzi, Thomas Niehaus, Mario Barbatti, and Walter Thiel,
   J. Chem. Theory Comput., 2017, 13, pp 515-524.
2. Sandro Giuseppe Chiodo, Monica Leopoldini, MolSOC: A spin-orbit coupling code,
   Computer Physics Communications, 2014, 185, pp 676-683.

Energies from QM calculation:
-------------------------------------------------------------
Singlet(eV):      4.0272      8.4726      9.2789      9.5754
Triplet(eV):      3.3510      5.6721      8.0749      8.1309
-------------------------------------------------------------

SOC information from PySOC calculation:
-------------------------------------------------------------------------------------------------------
sum_soc, <S0|Hso|T1,1,0,-1> (cm-1):        60.74192       42.95102        0.00769       42.95102
sum_soc, <S0|Hso|T2,1,0,-1> (cm-1):         0.01943        0.01374        0.00001        0.01374
sum_soc, <S0|Hso|T3,1,0,-1> (cm-1):        10.62339        0.01332       10.62338        0.01332
sum_soc, <S0|Hso|T4,1,0,-1> (cm-1):        59.88731       42.34673        0.00124       42.34673
sum_soc, <S1|Hso|T1,1,0,-1> (cm-1):         0.00085        0.00060        0.00000        0.00060
sum_soc, <S1|Hso|T2,1,0,-1> (cm-1):        44.31445       31.33504        0.02237       31.33504
sum_soc, <S1|Hso|T3,1,0,-1> (cm-1):         8.62802        6.10093        0.00018        6.10093
sum_soc, <S1|Hso|T4,1,0,-1> (cm-1):        50.72112        0.01511       50.72112        0.01511
sum_soc, <S2|Hso|T1,1,0,-1> (cm-1):         7.38543        5.22229        0.00015        5.22229
sum_soc, <S2|Hso|T2,1,0,-1> (cm-1):         0.25315        0.00080        0.25315        0.00080
sum_soc, <S2|Hso|T3,1,0,-1> (cm-1):         0.00029        0.00020        0.00002        0.00020
sum_soc, <S2|Hso|T4,1,0,-1> (cm-1):         0.24323        0.17198        0.00126        0.17198
sum_soc, <S3|Hso|T1,1,0,-1> (cm-1):        51.25782        0.03351       51.25780        0.03351
sum_soc, <S3|Hso|T2,1,0,-1> (cm-1):        37.26932       26.35339        0.00077       26.35339
sum_soc, <S3|Hso|T3,1,0,-1> (cm-1):         1.42436        1.00717        0.00581        1.00717
sum_soc, <S3|Hso|T4,1,0,-1> (cm-1):         0.05572        0.03940        0.00000        0.03940
sum_soc, <S4|Hso|T1,1,0,-1> (cm-1):         5.85744        4.14184        0.00012        4.14184
sum_soc, <S4|Hso|T2,1,0,-1> (cm-1):         0.23047        0.00177        0.23046        0.00177
sum_soc, <S4|Hso|T3,1,0,-1> (cm-1):         0.00100        0.00071        0.00006        0.00071
sum_soc, <S4|Hso|T4,1,0,-1> (cm-1):         0.94719        0.66976        0.00094        0.66976
-------------------------------------------------------------------------------------------------------
[EOF]
```

There are four numbers in each line corresponding respectively to root sum square of the subshell number, the module length of the subshell with quantum numbers 1, 0, -1. The unit is in `cm-1`. For example, the `<S0|Hso|T1,1,0,-1>` means SOC between ground state and first triplet with quantum numbers 1, 0, -1, respectively.