PySOC

Calculation of Spin-Orbit Coupling

Quick Reference Guide

Version 2.1.4

16 February 2021

Oliver S. Lee

Table of Contents

How to Use This Document

This document is designed to act as rapid introduction and quick-reference to the PySOC spin-orbit coupling calculation program.

Images are included for comparison and reference.

Supplementary information and helpful tips are displayed separately, as follows:

NOTE: This is a tip.

Commands that should be typed by the user are displayed in the following format:

```
> echo Hello world
```

The arrow character (>) should not be typed; it is used to differentiate commands that should be typed from the resulting output (which will be displayed without the arrow character):

```
> echo Hello world
Hello world
```

In the above example, the user is being instructed to type: echo Hello world. The computer, in response, gives the output: Hello world.

Ellipses (…) indicate that the real, full output has been truncated:

```
> cat /etc/fstab
# /etc/fstab: static file system information.
#
...
```

About PySOC

PySOC is a program for the calculation of spin-orbit coupling (SOC) values between singlet and triplet excited states. As input, PySOC requires an excited states calculation at any of the following levels of theory:

- Time-dependant DFT (TD-DFT)[1–7] with Gaussian 09/16[8,9]
- TD-DFT using the Tamm–Dancoff approximation (TDA)[1–7,10] with Gaussian 09/16[8,9]
- Time-dependant Density Functional based Tight Binding (TD-DFTB) with DFTB+[11]

PySOC consists of two Fortran programs (molsoc and soc_td) and a Python controller (pysoc) with contributions from the following authors:

pysoc: Originally written by Xing Gao *et al.*[12] Rewritten for Python3.6 by the current author.

molsoc: Originally written by Sandro G. Chiodo and Monica Leopoldini,[13] with modifications by Xing Gao *et al.*[12]

soc_td: Originally written by Xing Gao et al,[12] with modifications by the current author.

Prerequisites

The four following prerequisites must be installed before PySOC can be used:

- **cclib:** The cclib parsing library,[14] required to parse results from completed Gaussian calculations. The user is directed to the cclib documentation for more information: https://cclib.github.io/.
- **tabulate:** Prints SOC results in a user-friendly table format.
- **scipy:** Used to provide certain scientific constants for the interconversion of energy units:
- **periodictable:** Interconverts between atomic number and symbol.

These requirements are most easily satisfied using the pip command:

```
> pip3 install --user cclib tabulate scipy periodictable
```

Installation

The latest version of PySOC can be downloaded from github with the git program:

```
> git clone https://github.com/oliver-s-lee/pysoc.git
```

This will create a ‘pysoc’ folder in the current directory in which PySOC will be downloaded. Within this folder, two further steps are required to ‘install’ PySOC:

- The ‘bin’ subdirectory should be added to the user’s PATH variable.
- The ‘pysoc’ top directory should be added to the user’s PYTHONPATH variable.

Installation can be made permanent by adding these paths to the user’s .bashrc file (located in the user’s home directory), for example:

```
# Install PySOC
PYTHONPATH="$PYTHONPATH:/path/to/pysoc”
PATH="$PATH:/path/to/pysoc/bin"
```

Installation can be tested with the ‘pysoc -v’ command, which will print the PySOC version if installation was successful:

```
> pysoc –v
2.1.4
```

Using PySOC

Setup

Before spin-orbit coupling can be calculated, it is first necessary to perform an excited states calculation with either the Gaussian or DFTB+ quantum mechanical (QM) calculation programs. The next section will briefly cover the necessary options to perform SOC calculations with each, but the reader is expected to be familiar with the basic operation of the QM program of choice.

NOTE: Shells higher than the f shell are not currently supported by PySOC. The user should ensure that the basis set used in the QM program does not contain g (or higher) shells.

Gaussian

Time-dependant DFT excited states calculations both with and without the Tamm–Dancoff approximation are supported for Gaussian versions 09 and 16. Older versions of Gaussian and alternative calculation methods (CIS etc.) may additionally be supported, but have not been tested.

PySOC requires that the following options be specified in the input file:

Option	Type	Description
%rwf=file.rwf	Link 0 option	Indicates that the read-write intermediate file should be created with the name ‘file.rwf’. PySOC expects the rwf file to have the same name as the log file (except for the extension), but this can be specified manually to pysoc if a different name is used.
6D	Route keyword	Specifies that pure (rather than Cartesian) d functions should be used.
10F	Route keyword	Specifies that pure (rather than Cartesian) f functions should be used.
GFInput	Route keyword	Specifies that the basis set should be printed in the calculation output.

In addition, either the ‘TD’ or ‘TDA’ keywords should be given to calculate excited states. An exampled input file is given below:

```
%mem=1GB
%chk=gaussian.chk
%rwf=gaussian.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
```

DFTB+

Please note that this section is under revision.

NOTE: PySOC requires Gaussian type orbitals (GTOs) to perform the SOC calculation, yet DFTB+ uses Slater-type orbitals (STOs). Thus the parameter set used for the DFTB+ calculation must be fitted to GTOs for use with PySOC. PySOC contains a fitted set for the mio-1-1 parameter set which will be used by default, alternative fitted sets can be specified with the '--fitted_basis' option:

```
> pysoc ch2o.xyz --fitted_basis /path/to/directory
```

An example dftb_in.hsd file is given below:

```
Geometry = GenFormat {
  <<< "ch2o.gen"
}

Driver = {}

Hamiltonian = DFTB {
  SCC = Yes
  SCCTolerance = 1e-10  # Very tight for test purposes only
  MaxAngularMomentum = {
    H  = "s"
    C  = "p"
    O  = "p"
  }
  SlaterKosterFiles = Type2FileNames {
    Prefix =
"/fsnfs/users/xinggao/work/gsh/thiothymine/gtsh/test_python/tddftb/sk/
mio-1-1/"
    Separator = "-"
    Suffix = ".skf"
  }
  LinearResponse {
    NrOfExcitations = 10
    StateOfInterest = 0
    Symmetry = both
    HubbardDerivatives {
      H  = 0.347100   0.491900
      C  = 0.341975   0.387425
      O  = 0.467490   0.523300
    }
    WriteTransitions = Yes
    WriteTransitionDipole = Yes
    WriteXplusY = Yes
  }
}

Options {
  WriteEigenvectors = Yes
  WriteHS = No
}
```

```
ParserOptions {
  ParserVersion = 4
}
```

After performing the DFTB+ calculation, set WriteHS = Yes and run the calculation a second time.

Spin-Orbit Coupling Calculation

Once the QM calculation of choice has completed successfully, SOC can be calculated using the 'pysoc' program. This program has one mandatory argument which is the principle output file from the completed QM calculation. For Gaussian, this is the .log file, while for DFTB+ this is the .xyz geometry file. PySOC will automatically find the additional input files it requires from the location of the .log or .xyz file.

NOTE: When computing SOC from a Gaussian calculation, the location of the .rwf file can be given explicitly using the '--rwf_file' argument:

```
> pysoc ch2o.log --rwf_file gaussian.rwf
```

For example, to calculate SOC for a file in the current directory with the name ch2o.log:

```
> pysoc ch2o.log
```

Or to calculate SOC from a completed DFTB+ calculation:

```
> pysoc ch2o.xyz
```

Each calculation has a typical duration of a few minutes, and the output will be presented in a user-friendly table format:

```
> pysoc ch2o.log
 Singlet    Triplet     RSS (cm-1)    +1 (cm-1)    0 (cm-1)    -1 (cm-1)
---------  ---------  ------------  -----------  ----------  -----------
   S0         T1          60.7419      42.9510      0.0077      42.9510
   S0         T2           0.0194       0.0137      0.0000       0.0137
   S0         T3          10.6234       0.0133     10.6234       0.0133
   S0         T4          59.8873      42.3467      0.0012      42.3467
   S0         T5          11.7332       0.0013     11.7332       0.0013
   S1         T1           0.0008       0.0006      0.0000       0.0006
   S1         T2          44.3144      31.3350      0.0224      31.3350
   S1         T3           8.6280       6.1009      0.0002       6.1009
   S1         T4          50.7211       0.0151     50.7211       0.0151
   S1         T5           5.5317       3.9115      0.0001       3.9115
   S2         T1           7.3854       5.2223      0.0001       5.2223
   S2         T2           0.2532       0.0008      0.2531       0.0008
   S2         T3           0.0003       0.0002      0.0000       0.0002
   S2         T4           0.2432       0.1720      0.0013       0.1720
 ...
```

Here, each line indicates SOC between one singlet and one triplet state, which are given by the 'Singlet' and 'Triplet' columns respectively. The last three columns each contain the calculated spin-orbit coupling with quantum numbers +1, 0 and -1 respectively for those two states, which are summarised by the RSS column containing the root sum square of these three values. All SOC values are given in wavenumbers (cm^{-1}).

Alternatively, a comma-separated values (CSV) format can be requested with the '-c' option, which can be more easily imported into a spreadsheet for analysis:

```
> pysoc ch2o.log -c
Singlet,Triplet,RSS (cm-1),+1 (cm-1),0 (cm-1),-1 (cm-1)
S0,T1,60.7419154885397,42.95102,0.00769,42.95102
S0,T2,0.019431296920174937,0.01374,1e-05,0.01374
S0,T3,10.623396701112124,0.01332,10.62338,0.01332
S0,T4,59.887319900989056,42.34673,0.00124,42.34673
S0,T5,11.733160146260683,0.00131,11.73316,0.00131
...
```

To write to a file instead of to the screen, use standard Linux file redirection with the '>' or '>>' characters (to overwrite or append to an existing file respectively):

```
> pysoc ch2o.log -c > SOC.csv
```

Intermediate Files

PySOC generates a number of intermediate files during operation. By default, these are deleted at the end of the calculation and only the final SOC values are displayed. To override this behaviour and keep these intermediate files, use the '-o' (output) option with a path to a directory where the intermediate files should be stored. One may use '-o ./' to write the intermediate files to the current directory. The '-o' option has no effect on the final SOC values:

```
> pysoc ch2o.log -o ./
 Singlet    Triplet     RSS (cm-1)    +1 (cm-1)    0 (cm-1)    -1 (cm-1)
---------  ---------  ------------  -----------  ----------  -----------
   S0         T1           60.7419      42.9510      0.0077      42.9510
   S0         T2            0.0194       0.0137      0.0000       0.0137
   S0         T3           10.6234       0.0133     10.6234       0.0133
   S0         T4           59.8873      42.3467      0.0012      42.3467
   S0         T5           11.7332       0.0013     11.7332       0.0013
...
```

Referencing PySOC

Scientific publications which make use of PySOC in their work should cite the original PySOC publication by Xing Gao *et al.* as well as the cclib publication by O'Boyle *et al.*:

Evaluation of Spin-Orbit Couplings with Linear-Response Time-Dependent Density Functional Methods, Xing Gao, Shuming Bai, Daniele Fazzi, Thomas Niehaus, Mario Barbatti, and Walter Thiel, *J. Chem. Theory Comput.*, 2017, **13** (2), 515–524. DOI: 10.1021/acs.jctc.6b00915

cclib: A Library for Package-Independent Computational Chemistry Algorithms, N. M. O'Boyle, A. L. Tenderholt and K. M. Langner, *J. Comput. Chem.*, 2008, **29**, 839–845. DOI: 10.1002/jcc.20823

Bibliography

1 R. Bauernschmitt and R. Ahlrichs, *Chem. Phys. Lett.*, 1996, **256**, 454–464.

2 M. E. Casida, C. Jamorski, K. C. Casida and D. R. Salahub, *J. Chem. Phys.*, 1998, **108**, 4439–4449.

3 G. Scalmani, M. J. Frisch, B. Mennucci, J. Tomasi, R. Cammi and V. Barone, *J. Chem. Phys.*, 2006, **124**, 094107.

4 C. Van Caillie and R. D. Amos, *Chem. Phys. Lett.*, 2000, **317**, 159–164.

5 F. Furche and R. Ahlrichs, *J. Chem. Phys.*, 2002, **117**, 7433–7447.

6 C. Van Caillie and R. D. Amos, *Chem. Phys. Lett.*, 1999, **308**, 249–255.

7 R. E. Stratmann, G. E. Scuseria and M. J. Frisch, *J. Chem. Phys.*, 1998, **109**, 8218–8224.

8 M. J. Frisch, G. W. Trucks, H. B. Schlegel, G. E. Scuseria, M. A. Robb, J. R. Cheeseman, G. Scalmani, V. Barone, B. Mennucci, G. A. Petersson, H. Nakatsuji, M. Caricato, X. Li, H. P. Hratchian, A. F. Izmaylov, J. Bloino, G. Zheng, J. L. Sonnenberg, M. Hada, M. Ehara, K. Toyota, R. Fukuda, J. Hasegawa, M. Ishida, T. Nakajima, Y. Honda, O. Kitao, H. Nakai, T. Vreven, J. A. Montgomery, J. E. Peralta, F. Ogliaro, M. Bearpark, J. J. Heyd, E. Brothers, K. N. Kudin, V. N. Staroverov, T. Keith, R. Kobayashi, J. Normand, K. Raghavachari, A. Rendell, J. C. Burant, S. S. Iyengar, J. Tomasi, M. Cossi, N. Rega, J. M. Millam, M. Klene, J. E. Knox, J. B. Cross, V. Bakken, C. Adamo, J. Jaramillo, R. Gomperts, R. E. Stratmann, O. Yazyev, A. J. Austin, R. Cammi, C. Pomelli, J. W. Ochterski, R. L. Martin, K. Morokuma, V. G. Zakrzewski, G. A. Voth, P. Salvador, J. J. Dannenberg, S. Dapprich, A. D. Daniels, Farkas, J. B. Foresman, J. V Ortiz, J. Cioslowski and D. J. Fox, *Gaussian 09, Revis. D.01, Gaussian, Inc., Wallingford CT*, 2013.

9 M. J. Frisch, G. W. Trucks, H. B. Schlegel, G. E. Scuseria, M. A. Robb, J. R. Cheeseman, G. Scalmani, V. Barone, G. A. Petersson, H. Nakatsuji, X. Li, M. Caricato, A. V Marenich, J. Bloino, B. G. Janesko, R. Gomperts, B. Mennucci, H. P. Hratchian, J. V Ortiz, A. F. Izmaylov, J. L. Sonnenberg, D. Williams-Young, F. Ding, F. Lipparini, F. Egidi, J. Goings, B. Peng, A. Petrone, T. Henderson, D. Ranasinghe, V. G. Zakrzewski, J. Gao, N. Rega, G. Zheng, W. Liang, M. Hada, M. Ehara, K. Toyota, R. Fukuda, J. Hasegawa, M. Ishida, T. Nakajima, Y. Honda, O. Kitao, H. Nakai, T. Vreven, K. Throssell, J. A. Montgomery Jr., J. E. Peralta, F. Ogliaro, M. J. Bearpark, J. J. Heyd, E. N. Brothers, K. N. Kudin, V. N. Staroverov, T. A. Keith, R. Kobayashi, J. Normand, K. Raghavachari, A. P. Rendell, J. C. Burant, S. S. Iyengar, J. Tomasi, M. Cossi, J. M. Millam, M. Klene, C. Adamo, R. Cammi, J. W. Ochterski, R. L. Martin, K. Morokuma, O. Farkas, J. B. Foresman and D. J. Fox, *Gaussian 16, Revis. C.01, Gaussian, Inc., Wallingford CT*, 2016.

10 S. Hirata and M. Head-Gordon, *Chem. Phys. Lett.*, 1999, **314**, 291–299.

11 B. Hourahine, B. Aradi, V. Blum, F. Bonafé, A. Buccheri, C. Camacho, C. Cevallos, M. Y. Deshaye, T. Dumitrică, A. Dominguez, S. Ehlert, M. Elstner, T. van der Heide, J. Hermann, S. Irle, J. J. Kranz, C. Köhler, T. Kowalczyk, T. Kubař, I. S. Lee, V. Lutsker, R. J. Maurer, S. K. Min, I. Mitchell, C. Negre, T. A. Niehaus, A. M. N. Niklasson, A. J. Page, A. Pecchia, G. Penazzi, M. P. Persson, J. Řezáč, C. G. Sánchez, M. Sternberg, M. Stöhr, F. Stuckenberg, A. Tkatchenko, V. W. Z. Yu and T. Frauenheim, *J. Chem. Phys.*, 2020, **152**, 124101.

12 X. Gao, S. Bai, D. Fazzi, T. Niehaus, M. Barbatti and W. Thiel, *J. Chem. Theory Comput.*, 2017, **13**, 515–524.

13 S. G. Chiodo and M. Leopoldini, *Comput. Phys. Commun.*, 2014, **185**, 676–683.

14 N. M. O'Boyle, A. L. Tenderholt and K. M. Langner, *J. Comput. Chem.*, 2008, **29**, 839–845.