pyVibLum User Manual

Version 1.0

About pyVibLum

pyVibLum calculates the vibronic structure of the lanthanide emission coupled to molecular vibrations and predicts the electron-vibrational coupling strengths for specific vibrations.

For user support, contact svarganov@unr.edu.

License

Citing pyVibLum

If you use pyVibLum, please cite

V. D. Dergachev, L. F. Chibotaru, S. A. Varganov, *J. Phys. Chem. Lett.* 2025, 16, 2309-2313
DOI: 10.1021/acs.jpclett.4c03531

Table of content

Installation

Installation via conda

We recommend installing pyVibLum via conda to avoid interference with existing environment and to resolve all the dependencies (NumPy, SciPy). Installation requires Python >= 3.0.

These are the steps to install and use pyVibLum via conda:

1. Download pyVibLum from GitHub: https://github.com/svarganov/pyVibLum

2. Create 'MyEnv' conda environment:

   ```
   % conda create -n MyEnv python=3.6
   % conda activate MyEnv
   ```

   If using Miniconda, a Python interpreter will be installed in

   ```
   /home/user/miniconda3/envs/MyEnv/bin/python
   ```

   A default directory for Python packages installation will be created:

   ```
   /home/user/miniconda3/envs/MyEnv/lib/python3.6/site-packages
   ```

3. Under activated 'MyEnv', in the top directory (containing `setup.py`), execute:

   ```
   % python -m pip install .
   ```

Now, pyVibLum is installed in `site-packages` and accessible when the 'MyEnv' environment is activated.

To deactivate conda's environment, execute

```
% conda deactivate
```

Installation of conda

In case conda is not installed, the following installation steps can prove useful:

1. Download conda source. For example:

   ```
   % wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh  -O
   miniconda.sh
   ```

2. Run miniconda.sh script and follow the installation steps:

   ```
   % bash miniconda.sh
   ```

3. Perform post-installation steps (advised during installation):

   ```
   % source ~/miniconda3/etc/profile.d/conda.sh
   % conda config --set auto_activate_base false
   % conda tos accept --override-channels --channel https://repo.anaconda.com/pkgs/main
   % conda tos accept --override-channels --channel https://repo.anaconda.com/pkgs/r
   ```

The above steps only serve as example and do not bypass the official installation guide.

Theory

pyVibLum implements methodology which describes electrodipolar transitions in lanthanides coupled to molecular vibrations. This methodology predicts the vibronic structure associated with the static mechanism of the emission, which is dominant in systems without inversion symmetry. Because of the crystal field asymmetry, the parity forbidden transitions between the electronic states of the 4*f* subshell become allowed due to the admixing of the orbitals of the opposite parity into wavefunctions of these states.

The central quantities of this methodology are the electron-vibrational coupling strengths and intensity of vibronic emission from the emitting state *e* to ground state *g*. The intensity is calculated using the Fermi's golden rule:

$$I(\Omega) = \frac{2\pi}{\hbar}|\mu_{el}|^2 \sum_{n_1=0}^{\infty}\sum_{n_2=0}^{\infty}\dots\sum_{n_\mu=0}^{\infty}\dots\left(\prod_{\mu}\frac{1}{n_\mu!}S_\mu^{n_\mu}e^{-S_\mu}\right)\times$$
$$\times\,\delta\left(E^e_{\min}-E^g_{\min}-\hbar n_1\omega_1-\hbar n_2\omega_2-\dots-\hbar\Omega\right).$$

Here,

$\hbar\Omega$ – energy of the emitted photon,
μ_{el} – matrix element of the transition electric dipole moment between states *i* and *f*,
n_μ – number of vibrational quanta in mode μ,
S_μ – strength of electron-vibrational coupling with respect to mode μ,
$E_{\min}$ – state energy at its equilibrium geometry,
ω – vibrational frequency.

Calculation of the coupling strengths S_μ requires gradients of the emitting and ground spin-orbit states along normal modes of vibration. These gradients are expressed via the gradients of the spin-free states and their corresponding nonadiabatic coupling matrix elements. This determines the input data required for pyVibLum.

Input data

A pyVibLum calculation requires data from the electronic structure calculations:

1. XYZ coordinates at the reference, i.e., DFT-optimized geometry
2. Molecular Hessian at the reference geometry
3. Energies of the spin-orbit states
4. Oscillator strengths between the emitting and ground spin-orbit states
5. Expansion coefficients of the spin-orbit states in the basis of spin-free CASSCF electronic states
6. Gradients of the CASSCF states
7. Nonadiabatic coupling (NAC) matrix elements between the CASSCF states

pyVibLum is automated to read many of these data with only little help from the user.

Collecting the data

1. XYZ coordinates should be saved in the common XMOL format:

```
number of atoms
comment line
atom1          x1       y1       z1
atom2          x2       y2       z2
...
atomN          xN       yN       zN
```

For example,

```
62
symmetry c1
Er      6.506058781     3.732013885     3.307242557
O       4.665931933     3.991432931     2.152470270
N       4.816779928     2.238604365     4.391861806
N       6.515436318     3.729413086     6.245285784
C       3.521969010     3.421536311     1.917327952
...
H       7.488284608     0.345768331     0.239159961
```

2. pyVibLum reads molecular hessians obtained from the following electronic structure codes:
   - ORCA (.hess)
   - GAMESS (.dat)
   - Q-Chem (.fchk)

3. The electronic structure (energies, osc. strengths, spin-orbit coefficients, CASSCF gradients and NACs) is to be obtained from OpenMolcas.

The energies, osc. strengths, and spin-orbit coupling coefficients are read by pyVibLum from the OpenMolcas output file.

For the energies, look for the &RASSI output section:

```
# OpenMolcas output file
                                        &RASSI
...
          ************************************************
          *
          *                    Spin-orbit section
          *
          ************************************************

     Total energies including SO-coupling:
```

The osc. strengths can be found farther down the output file:

```
# OpenMolcas output file
++ Dipole transition strengths (SO states):
-----------------------------------------------------------
...
From   To       Osc. strength     Einstein coefficients Ax, Ay, Az (sec-1)    Total A (sec-1)
```

The spin-orbit states are written in the basis of the spin-free CASSCF states:

$$\Psi_r = \sum_{N,S,M_S} c^r_{N,S,M_S} \Psi_{N,S,M_S},$$

where *N* is an index of an electronic (CASSCF) state, *S* is the total electron spin, M_S is the spin projection, and c^r_{N,S,M_S} are the complex-valued expansion coefficients.

OpenMolcas output file

Complex eigenvectors in basis of non-so eigenstates:

…

Energy (au)			-0.03314576	-0.03314576	-0.03274730	-0.03274730
SFS	S	Ms	1	2	3	4
1	1.5	-1.5	(-0.3301, 0.5034)	(-0.0034, 0.0095)	(0.0433,-0.0295)	(0.0204,-0.0569)
1	1.5	-0.5	(0.0093,-0.0009)	(-0.0324, 0.0312)	(0.0566,-0.0399)	(0.0250, 0.2373)
…						
5	1.5	0.5	(0.0277,-0.0094)	(0.0055, 0.0174)	(0.2363, 0.1062)	(-0.0383,-0.0165)
5	1.5	1.5	(-0.0350, 0.0143)	(-0.1094,-0.0293)	(-0.1361, 0.0503)	(0.1180,-0.1205)

…

The c^r_{N,S,M_S} coefficients are therefore stored as in columns for every spin-orbit state *r*.

The gradients of the spin-orbit states, which are required to calculate the electron-vibrational coupling strength and vibronic intensities, can be obtained from the CASSCF gradients and NAC matrix elements between the CASSCF states. The OpenMolcas output files containing gradients and NACs should be stored in the path_to_grads and path_to_nacs directories (Input Structure section):

Directory containing gradients

gradient1.out gradient2.out gradient3.out … gradientN.out …

Directory containing NACs

nac_1_2.txt nac_1_3.txt nac_1_4.txt … nac_3_4.txt nac_3_5.txt …

Please note that the files containing the gradients should be explicitly named 'gradientX', where X is the index of the CASSCF state. Indices should start with 1. The file extension is arbitrary and can be omitted.

Please note that the files containing the NAC matrix elements can be named arbitrarily. For convenience, these were labeled as nac_N1_N2.txt, where N1 and N2 are indices of the CASSCF states. The file extension is arbitrary and can be omitted.

Notes on the OpenMolcas calculations

1. Printing thresholds. The osc. strengths are printed only if larger than a selected threshold. If an osc. strength for a particular transition is missing in the output file, it will be substituted by 0.0. In this case, the vibronic structure cannot be resolved for that transition. We recommend to change the OpenMolcas DIPRint and TDMN thresholds and re-run the electronic structure calculations.

   The printing level in OpenMolcas is controlled by the MOLCAS_PRINT environment variable, which is by default set to 2. Printing the spin-orbit expansion coefficients requires MOLCAS_PRINT=3, which can be set by executing export MOLCAS_PRINT=3 in the terminal or job submission script. The default MOLCAS_PRINT=2 is sufficient for the CASSCF gradients and NACs calculations.

   ! A remark (optional): to keep only necessary part of the OpenMolcas output for the pyVibLum calculation, and for better navigation in this output, authors made trimmed copies of the output files (examples/data/{OpenMolcas.out, gradients, nacs}). If it sounds appealing to the user, please be cautious to keep the following part in the output:

   - In OpenMolcas.out, all the output before the &RASSI section can be deleted.
   - In gradients/gradientN.out, all the output before the line containing Molecular gradients can be deleted.
   - In nacs/nac_N1_N2.txt, all the output before the line containing Lagrangian multipliers are calculated for states no. N1/ N2 can be deleted.

   We recommend the user to check the example OpenMolcas output files for clarity.

2. NACs. The NAC matrix elements need to be calculated for only the N2 > N1 pairs of states, since the coupling operator is Hermitian and the CASSCF wavefunctions are real. That is, for a total of N_{total} CASSCF states, only $[(N_{total})^2 - N_{total}]/2$ matrix elements need to be calculated. In a simple example for, say, 4 CASSCF states, there is a total of six matrix elements required: NAC_1_2, NAC_1_3, NAC_1_4, NAC_2_3, NAC_2_4, and NAC_3_4.

Examples of the OpenMolcas input files are given in the **Electronic structure input**.

Input structure

pyVibLum accepts input in the form of the Python script, i.e., start.py, which can be found in examples. First, paths to the input data need to be specified. Check the following example:

```
# from start.py
base_path = Path.cwd()
subdir = "data"

path_to_elec  = base_path.joinpath(subdir,"OpenMolcas.out")
path_to_grads = base_path.joinpath(subdir,"gradients")
path_to_nacs  = base_path.joinpath(subdir,"nacs")
path_to_coord = base_path.joinpath(subdir,"coord.xyz")
path_to_hess  = base_path.joinpath(subdir,"hessian.dat")
```

Here, Path.cwd() sets the path to the current working directory, i.e., /home/user/run_pyVibLum/my_example. It is then assumed that all the data is stored in the "data" subdirectory: /home/user/run_pyVibLum/my_example/data. Then, the path_ variables expand to

```
path_to_elec  = /home/user/run_pyVibLum/my_example/data/OpenMolcas.out
path_to_grads = /home/user/run_pyVibLum/my_example/data/gradients
path_to_nacs  = /home/user/run_pyVibLum/my_example/data/nacs
path_to_coord = /home/user/run_pyVibLum/my_example/data/coord.xyz
path_to_hess  = /home/user/run_pyVibLum/my_example/data/hessian.dat
```

The paths to data can be changed for user's convenience; not all of the data need to reside within a single subdirectory, i.e. "data". For example, individual paths can be set, and then base_path and subdir are no longer needed:

```
path_to_elec  = /my/path/to/electronic/myMolcasName.out
path_to_grads = /my/path/to/gradients/MyGradientsDirectory
path_to_nacs  = /my/path/to/nacs/MyNACsDirectory
path_to_coord = /my/path/to/coordinates/MyCoord.xyz
path_to_hess  = /my/path/to/hessian/MyHessian(.dat, .hess, or .fchk)
```

Next, user needs to specify the CASSCF electronic states used in the electronic structure calculations. As example, consider the $^4F_{9/2}$ to $^4I_{15/2}$ emission in the erbium trensal complex. Due to the electron-electron interaction, the $^4f^{11}$ configuration of atomic erbium gives rise to the ^{4}I, ^{4}F, and ^{4}S low-lying atomic terms comprising a total of 21 electronic quartet states (S = 3/2 is spin not the term symbol). All 21 states must be included in the OpenMolcas spin-orbit coupling calculation, and hence, all the 21 states should be given in start.py, for example, using Python list:

```
# from start.py
states_electronic = list(range(1, 22, 1))
# expands to [1, 2, 3, …, 20, 21]
```

Next, the spin-orbit states are to be provided. In this example, we consider emission from the lowest-energy state of the $^4F_{9/2}$ multiplet to all the states of the ground $^4I_{15/2}$ multiplet. The total number of the spin-orbit states that arise from an atomic term is (2L+1)(2S+1), where L and S are the orbital and spin angular momenta. For the ^{4}I term, L = 6, and hence, there are 13*4 = 52 spin-orbit states, which form the $^4I_{15/2}$, $^4I_{13/2}$, $^4I_{11/2}$, and $^4I_{9/2}$ multiplets. For the ^{4}F term, L = 3 giving 7*4 = 28 spin-orbit states, which comprise the $^4F_{9/2}$, $^4F_{7/2}$, $^4F_{5/2}$, and $^4F_{3/2}$ multiplets. The S term is orbitally non-degenerate giving a single $^4S_{3/2}$ multiplet of the total of 4 spin-orbit states. In Er^{3+}, the energy order is $^4I_{15/2}$, $^4I_{13/2}$, $^4I_{11/2}$, $^4I_{9/2}$, $^4F_{9/2}$, $^4S_{3/2}$, … Given that the spin-orbit states form Kramers' doublets, for the electron spin is half-integer (S = 3/2 for Er^{3+}), every state is doubly degenerate. For every doublet, we choose only a single component. Hence, for the ground multiplet $^4I_{15/2}$, which has a total of 2*15/2 + 1 = 16 states, we choose only eight states with unique energies: 1, 3, 5, 7, 9, 11, 13, and 15. The emitting states of the $^4F_{9/2}$ manifold start with the degenerate states 53 and 54. We choose the 53rd state as the emitting state. Therefore, the spin-orbit states to be included in pyVibLum are 1, 3, 5, 7, 9, 11, 13, 15, and 53. There are multiple options:

```
# option 1 (start.py)
states_spin_orbit = list(range(1, 17, 2))
states_spin_orbit.append(53)

# option 2
states_spin_orbit = list(range(1, 17, 2)) + [53]

# option 3
states_spin_orbit = [1, 3, 5, 7, 9, 11, 13, 15, 53]
```

The rest of the input is straightforward and is summarized in a table below. All the parameters are pre-set in the start.py script.

Parameter	Type	Description
path_to_elec	str	Path to the OpenMolcas output file
path_to_coord	str	Path to the XYZ molecular geometry
path_to_hess	str	Path to the molecular Hessian
path_to_grads	str	Path to the OpenMolcas CASSCF gradients
path_to_nacs	str	Path to the OpenMolcas CASSCF NACs
states_electronic	list	List of CASSCF states used in OpenMolcas calculation
states_spin_orbit	list	List of the spin-orbit states included in lanthanide emission
S	float	Spin multiplicity. Currently restricted to be the same for all states
hessian_format	str	Format type of the molecular Hessian. Supports ORCA, GAMESS, and Q-Chem
line_width	float	Number in cm^{-1}. Used to set linewidth of the Gaussian approximating the delta function
min_energy	float	Number in eV. Minimum value of the $\hbar\Omega$ energy of the emitted photon (scanning frequency)
max_energy	float	Number in eV. Maximum value of $\hbar\Omega$
step	float	Number in eV. Scanning frequency step
num_exct	int	Maximum number of excitations per vibrational mode (maximum value of $n_1 = n_2 = \ldots = n_\mu = n$) to be used in intensity calculation
num_modes	int	Maximum number of modes with largest coupling strengths to be included in intensity calculation

The parameters are passed to pyVibLum in the format of Python dictionary objects. The dictionary structures themselves need not to be changed by the user.

To run the script in the Linux command terminal, use something like

```
python start.py > myoutfile.out 2>&1
```

Output

pyVibLum creates two output files: text output (standard output) and the vibronic intensities saved in the plain text format (Spectrum.txt).

The standard output contains the information about the modes (set by num_modes) which contribute the most to the vibronic structure of the emission. This information is organized in tables which contain the modes' frequencies (cm^{-1}) and Huang-Rhys factors for every electronic transition.

The Spectrum.txt file contains three columns: scanning frequency (eV), vibronic intensity (a. u.), and total intensity (a.u.). The vibronic intensities are obtained by subtracting intensities of the pure electronic transitions ($n_1 = n_2 = \ldots n_\mu = \ldots = 0$) from the spectrum giving the vibronic structure of the emission.

Examples

Two examples of the pyVibLum calculation are given in the examples directory. These correspond to the XYZ geometry and molecular Hessian obtained in either GAMESS or ORCA electronic structure codes. The rest of the data – energies of the spin-orbit states, osc. strengths, and the spin-orbit coefficients – is the same for both examples. Please note that these data are obtained using the DFT-optimized geometry in GAMESS. The ORCA example serves only the demonstration purpose.

To run either example, only few keywords need to be changed in examples/start.py while keeping the rest of the file as is:

```
# Example 1 (GAMESS)

path_to_coord = base_path.joinpath(subdir,"GCoord.xyz")
path_to_hess  = base_path.joinpath(subdir,"GHessian.dat")

hessian_format = "gamess"

# Example 2 (ORCA)

path_to_coord = base_path.joinpath(subdir,"OCoord.xyz")
path_to_hess  = base_path.joinpath(subdir,"OHessian.dat")

hessian_format = "orca"
```

The reference output files are given in the examples/out subdirectory. We recommend checking your calculations against the reference once pyVibLum is installed.

Electronic structure input

The energies of the spin-orbit states, the osc. strengths, and the spin-orbit coefficients can be obtained in an OpenMolcas calculation using keywords given below. Please note that the input provided serves only as an example and should be customized for user's needs.

```
# Example of OpenMolcas input file

&GATEWAY
title = My OpenMolcas calculation
basis = ANO-RCC-VDZP
coord = mygeom.xyz
group = C1

RICD
AMFI
ANGM
x      y      z

&SEWARD

&RASSCF
Spin
4
nActel
11
Inactive
149
Ras2
7
CIRoot
21 21 1
End of Input

&RASSI
SpinOrbit
EJOB
```

Nr of Jobiphs
1 21
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21
DIPRint
1.0D-9
TDMN
1.0D-9
End Of Input

Input example for the CASSCF gradient calculations:

Example of OpenMolcas input file

&GATEWAY
title = My OpenMolcas calculation
basis = ANO-RCC-VDZP
coord = mygeom.xyz
group = C1
RICD

>>> Do while
&SEWARD
doanalytical

&RASSCF
Spin
4
nActel
11
Inactive
149
Ras2
7
CIRoot
21 21 1
Rlxroot=10
End of Input

&SLAPAF

```
ITER
1
End of Input
>>> EndDo
```

Input example for the NAC calculations; multiple calculations can be set within a single input file:

```
# Example of OpenMolcas input file

&GATEWAY
title = My OpenMolcas calculation
basis = ANO-RCC-VDZP
coord = mygeom.xyz
group = C1
RICD

&SEWARD
doanalytical

&RASSCF
Spin
4
nActel
11
Inactive
149
Ras2
7
CIRoot
21 21 1
Rlxroot=10
End of Input

&ALASKA
NAC = 1 2
End of Input

&ALASKA
```

NAC = 1 3
End of Input

&ALASKA
NAC = 1 4
End of Input