ElPh

Release 1.0.0

Lucas S. R. Cavalcante, Makena A. Dettmann, Corina A. Magdalen

Aug 19, 2022

CONTENTS

Welcome to ElPh's documentation! A workflow to compute electron phonon coupling in small molecules OSCs.
==========

CHAPTER

ONE

ELPH

Functions

`chdir`(folder)	Changes the working directory to folder if not already current wd.
`mkdir`(folder)	Creates folder in current wd unless OSError occurs.

CHAPTER

TWO

ELPH.JAVERAGE

Functions

compute_total_weight(centers_of_mass)	Computes the distances between the centers of mass.
get_centers_of_mass(atoms, n_components, ...)	Gets centers of mass for each molecule
get_javerage(pair)	Computes the transfer integral for a pair of molecules
is_triangle(centers_of_mass)	Checks whether centers of mass form a triangle and not a line Args: centers_of_mass (np.ndarray): An array containing the centers of mass of each molecule Returns: Bool: Whether the centers of mass form a triangle or not
javerage()	main function, calls unwrap atoms and get_javerage for each pair in the system.
unwrap_atoms([structure_file])	Unwraps the molecules and identifies pairs based on a structure file.
write_structure(label, component_list, ...)	Writes the structure to a file

CHAPTER

THREE

ELPH.SIGMA

Functions

displace_atom(atoms, ia, iv, sign, delta)	Displace one atomic position in the Atoms object
finite_dif([delta])	Compute Gaussian calculation for displaced system
get_deviation(pair_atoms, dj_av, temp)	Calculate standard deviation (sigma) of the transfer integral
get_displacements(atoms)	Returns displacement of each atom in each direction
get_dj_matrix(jlists, delta)	Matrix containing gradient of J, the transfer integral
get_sigma(pair[, delta, temp])	Calculate standard deviation of the transfer integral between a molecule pair using finite differences
load_phonons(pair_atoms[, phonon_file, map_file])	Loads phonon modes and returns frequencies, eigenvectors and number of q points
sigma()	Write phonon modes from phonopy result, and calculate the standard deviation (sigma) for each pair of molecules.

CHAPTER

FOUR

ELPH.MOLECULES

Classes

`Molecules`([nmuc, coordmol, unitcell, ...])

CHAPTER

FIVE

ELPH.PHONONS

Functions

write_phonons([mesh, phonopy_file])	Calculate phonon modes based on the force constants from phonopy and write phonon frequencies, displacements, and number of q points to numpy file.

CHAPTER

SIX

INSTALL

6.1 ElPh Requirements

- Gaussian
- Catnip

6.2 Installation on local machine

- Install ElPh:

```
pip install elph
```

- Install gaussian:

  Gaussian installation will vary group to group.

- Installing Catnip

  Download and install Docker

```
docker pull madettmann/catnip
```

- Set environment variables

  Add these lines to your configuration file (.bashrc). The following code uses example paths and must be edited according to your system.

```
export ELPH_CATNIP_CMD='docker run -i --rm -v $(pwd):/projects -u $(id -u):$(id -g)␣
↪madettmann/catnip'
export ASE_GAUSSIAN_COMMAND='Your Gaussian Command Here < PREFIX.com > PREFIX.log'
```

6.3 Installation on NERSC

There is no need to install the ElPh package and its dependencies. Just these lines to your configuration file (.bashrc) in your NERSC home folder.

```
module use /global/common/software/m2734/ElPh/modulefiles
module load elph
```

6.4 Usage

- Local machine

1. Add cif file to current working directory
2. Run javerage

   ```
   elph --javerage
   ```

3. Compute the phonon modes

   Refer to DCS-Flow for more information.

4. Run sigma

   Copy FORCE_SETS, phonopy_params.yaml to your working directory

   ```
   elph --sigma
   ```

5. Compute mobility

   Write input files and define correct parameters for the specific system

   ```
   elph --write_files
   ```

   Run mobility

   ```
   elph --mobility
   ```

6. Visualize sigma contribution per atom/mode

   To generate the visualization per atom:

   ```
   elph --view atoms
   ```

   To generate the visualization per mode (n highest modes):

   ```
   elph --view modes 3
   ```

- NERSC

The step-by-step is basically the same as in a local machine, however add these commands to the following steps

2. Once all the jobs submited by the Javerage command are done, run

   ```
   elph --read_javg
   ```

4. After all the jobs submited by the Sigma command are finished, use

```
elph --read_sigma
```

CHAPTER

SEVEN

EXAMPLES

7.1 Example 1: Workflow on local machine

The following example shows the complete workflow run on a local machine.

- Calculate transfer integral between pairs of molecules ($J_{average}$):

First, create a folder containing the geometry file (.cif, .gen, .sdf, or .xyz). The folder used in this example, named Anthracene, can be downloaded from the Uploads Folder.

In the Anthracene folder, unwrap the structure to whole molecules, and calculate the transfer integral between each unique pair of molecules in the system, using the following command.

```
elph --javerage
```

Once the job has completed, the following files and folders can be found in the Anthracene folder.

```
1/    A/    950158.cif        atom_mapping.json    J_C.json
2/    B/    all_pairs.json    J_A.json
3/    C/    all_pairs.xyz     J_B.json
```

The J files (J_A.json, J_B.json, J_C.json) present the transfer integral in meV of each pair described in all_pairs.json.

- Calculate the variance of transfer integrals (Sigma):

Before calculating Sigma, which is the variance of the transfer integral due to vibrations in the system, the phonons have to be computed. DCS-Flow calculates the phonon modes as the second part of its own workflow (2-phonons).

Copy the following files to the Anthracene folder

```
FORCE_SETS    phonopy_params.yaml
```

Calculate the variance (Sigma) within the finite differences method using the command

```
elph --sigma
```

After the job is done, the following files and folders will be written in the Anthracene folder.

```
1/displacements/...    A/displacements/...    A_disp_js.npz    Sigma_A.json    phonon.npz
2/displacements/...    B/displacements/...    B_disp_js.npz    Sigma_B.json
3/displacements/...    C/displacements/...    C_disp_js.npz    Sigma_A.json
```

The Sigma files (Sigma_A.json, Sigma_B.json, Sigma_C.json) present the variance of the transfer integral in meV of each pair

- Calculate the mobility

Create the lattice and parameters files, `lattice.json`and `params.json`, with the command

```
elph --write_files
```

Edit the files to match the following values

lattice.json:

```
{
   "nmuc": 2,
   "coordmol": [
      [0.0, 0.0, 0.0],
      [0.5, 0.5, 0.0]
      ],
   "unitcell": [
      [1.0, 0.0, 0.0],
      [0.0, 1.7321, 0.0],
      [0.0, 0.0, 1000.0]
   ],
   "supercell": [16, 16, 1],
   "unique": 6,
   "uniqinter": [
      [1, 1, 1, 0, 0, 1],
      [2, 2, 1, 0, 0, 1],
      [1, 2, 0, 0, 0, 3],
      [2, 1, 1, 0, 0, 2],
      [2, 1, 0, 1, 0, 2],
      [2, 1, 1, 1, 0, 3]
   ]
}
```

params.json:

```
{
   "javg": [0.058, 0.058, 0.058],
   "sigma": [0.029, 0.029, 0.029],
   "nrepeat": 50,
   "iseed": 3987187,
   "invtau": 0.005,
   "temp": 0.025
}
```

Use the following command to calculate the mobility (in $cm^2/(V \cdot s)$)

```
elph --mobility
```

- Visualize Sigma

In order to visualize the atomic contributions to Sigma, run

```
elph --view atoms
```

Or to visualize the 3 highest contributing phonon modes to Sigma, used

```
elph --view modes 3
```

7.2 Example 2: Workflow on NERSC

The following example shows the complete workflow run on the NERSC supercomputer.

- Calculate transfer integral between pairs of molecules ($J_{average}$):

Upload the Anthracene folder to NERSC using a file transfer software like Globus. The folder should contain the .cif file and the following run script (run.py)

```
#!/bin/bash
#SBATCH -J anthracene
#SBATCH -q debug
#SBATCH -N 1
#SBATCH -t 00:30:00
#SBATCH -C knl
#SBATCH --output=out.out
#SBATCH --error=err.out
#SBATCH --open-mode=append

#Print output before end of run
export PYTHONUNBUFFERED=1
export GAUSSIAN_BASIS='3-21G*'
export GAUSSIAN_CORES=12

#run the application:
eval $'elph --javerage'
```

Unwrap the structure to whole molecules, and submit the jobs to calculate the transfer integral between each unique pair of molecules in the system, by submitting the run script.

```
sbatch run.py
```

The script will submit a job for each molecule and pair of molecules (1, 2, 3, A, B, C). Check the progress of the submitted jobs using

```
sqs
```

Once all the jobs are done, read the calculated transfer integrals submitting the same run script changing the last line to

```
eval $'elph --read_javg'
```

After this, the following files and folders can be found in the Anthracene folder.

```
1/    A/    950158.cif        J_A.json    atom_mapping.json
2/    B/    all_pairs.json    J_B.json    err.out
3/    C/    all_pairs.xyz     J_C.json    out.out
```

The J files (J_A.json, J_B.json, J_C.json) present the transfer integral in meV of each pair described in all_pairs.json.

- Calculate the variance of transfer integrals (Sigma):

The phonons in the system have to be precomputed. DCS-Flow calculates the phonon modes as the second part of its own workflow (2-phonons).

Upload the following files to the Anthracene folder

```
FORCE_SETS    phonopy_params.yaml
```

Calculate the variance (Sigma) within the finite differences method submitting the run script with the updated command (depending on the size of the system 30 minutes in the debug queue is not enough to submit all the jobs, consider using -q as 'regular' and -t 01:00:00 or more)

```
eval $'elph --sigma'
```

A job for each displaced atom will be submitted. After all jobs are done, read the result submitting the run script with (this takes less than 10 minutes)

```
eval $'elph --read_sigma'
```

The following files and folders will be written in the Anthracene folder.

```
1/displacements/...    A/displacements/...    A_disp_js.npz    Sigma_A.json    phonon.npz
2/displacements/...    B/displacements/...    B_disp_js.npz    Sigma_B.json
3/displacements/...    C/displacements/...    C_disp_js.npz    Sigma_A.json
```

The Sigma files (Sigma_A.json, Sigma_B.json, Sigma_C.json) present the variance of the transfer integral in meV of each pair.

- Calculate the mobility

Create the lattice and parameters files, `lattice.json`and `params.json`, with the command

```
elph --write_files
```

Edit the files to match the following values

lattice.json:

```
{
   "nmuc": 2,
   "coordmol": [
      [0.0, 0.0, 0.0],
      [0.5, 0.5, 0.0]
      ],
   "unitcell": [
      [1.0, 0.0, 0.0],
      [0.0, 1.7321, 0.0],
      [0.0, 0.0, 1000.0]
   ],
   "supercell": [16, 16, 1],
   "unique": 6,
   "uniqinter": [
      [1, 1, 1, 0, 0, 1],
      [2, 2, 1, 0, 0, 1],
      [1, 2, 0, 0, 0, 3],
      [2, 1, 1, 0, 0, 2],
      [2, 1, 0, 1, 0, 2],
      [2, 1, 1, 1, 0, 3]
   ]
}
```

params.json:

```
{
   "javg": [0.058, 0.058, 0.058],
   "sigma": [0.029, 0.029, 0.029],
   "nrepeat": 50,
   "iseed": 3987187,
   "invtau": 0.005,
   "temp": 0.025
}
```

Submit the run script to calculate the mobility (in $cm^2/(V . s)$) by updating the command to

```
eval $'elph --mobility'
```

CHAPTER

EIGHT

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

e

INDEX

E

M