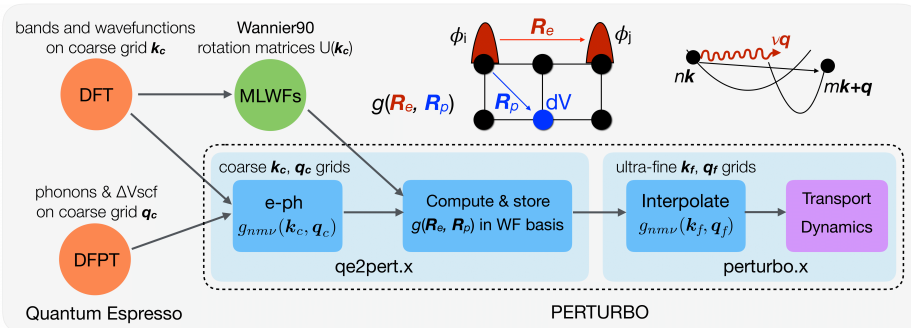
Perturbo

version 2.0

Last generated: July 01, 2022

Table of Contents

Introduction



PERTURBO is an open source software to compute from first principles the scattering processes between charge carriers (electrons and holes) and phonons, defects, and photons in solid state materials, including metals, semiconductors, oxides, and insulators. In the current version, PERTURBO mainly computes electron-phonon (e-ph) interactions and phonon limited transport properties in the framework of the Boltzmann transport equation (BTE). These include the carrier mobility, electrical conductivity, and Seebeck coefficient. PERTURBO can also compute the ultrafast carrier dynamics (for now, with fixed phonon occupations) by explicitly time-stepping the time-dependent BTE. We will include additional electron interactions, transport and ultrafast dynamics calculations in future releases.

PERTURBO is written in Fortran95 with hybrid parallelization (MPI and OpenMP) . The main output format is HDF5 , which is easily portable from one machine to another and is convenient for postprocessing using high-level languauges (e.g., Python). PERTURBO has a core software, called `perturbo.x`, for electron dynamics calculations and an interface software, called `qe2pert.x`, to read output files of Quantum Espresso (QE, version 6.4.1) and Wannier90 (W90, version 3.0.0 and higher). The `qe2pert.x` interface software generates an HDF5 file, which is then read from the core `perturbo.x` software. In principle, any other third-party density functional theory (DFT) codes (e.g., VASP) can use PERTURBO as long as the interface of the DFT codes can prepare an HDF5 output format for PERTURBO to read.

> For more details on the code, we refer the users to the manuscript accompying the source code:
> Jin-Jian Zhou, Jinsoo Park, I-Te Lu, Ivan Maliyov, Xiao Tong, Marco Bernardi, *"Perturbo: a software package for ab initio electron-phonon interactions, charge transport and ultrafast dynamics"*, *Comput. Phys.*

> *Commun.* **2021**, 107970

To download the code, contact us: perturbo@caltech.edu. For more information, please, visit the Download and installation section.

We gratefully acknowledge the National Science Foundation for supporting the development of PERTURBO.

Supported Features

The public version of PERTURBO has the following stable features:

- Phonon-limited carrier mobility, electrical conductivity and Seebeck coefficient
- Imaginary part of e-ph self-energy and e-ph scattering rates
- Phonon-limited carrier mean free path and relaxation times
- Magnetotransport calculations
- Ultrafast carrier dynamics with fixed phonon occupation
- Electron transport in the presence of high electric fields
- Calculations on magnetic systems with collinear spin
- Interpolated electronic band structure and phonon dispersion
- e-ph matrix elements for nonpolar and polar materials, and their Wannier interpolation
- Interface to TDEP for anharmonic phonons

All the calculations above can be done as a function of temperature and doping, for nonpolar and polar materials.

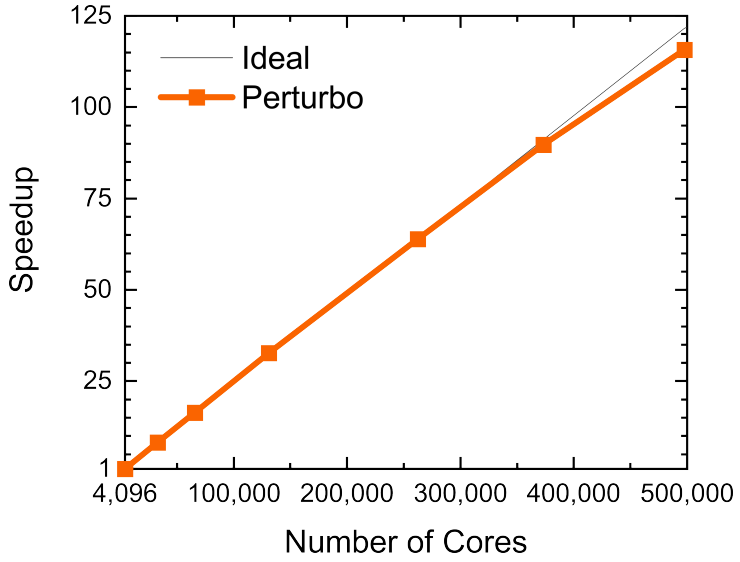
> ℹ **Note:** The current PERTURBO version supports both norm-conserving and ultrasoft pseudopotentials. Currently, the PAW pseudopotentials are not supported.

A brief summary of the PERTURBO calculation modes with the required input files can be found in the Interactive workflow section. For a detailed description of each calculation mode, please refer to the tutorial.

Code Performance and Scaling / Parallelization

This section discusses the scaling performance of the publicly available version of PERTURBO. Since its inception, PERTURBO implemented a hybrid MPI / OpenMP parallelization that allows for outstanding scaling on high-performance computing (HPC) platforms. To showcase the scaling performance, we present a calculation of the imaginary part of the electron-phonon self energy (calculation mode imsigma) in silicon using 72x72x72 electron **k**- and phonon **q**-point grids. The scaling test was performed using the Intel Xeon Phi 7250 Processors at the

National Energy Research Scientific Computing Center (NERSC). As seen from this figure, PERTURBO shows an almost linear scaling up to **500,000 cores** (the deviation from the linear scaling at 500,000 cores is less than 5%). This result, together with our ongoing work on the OpenACC GPU parallelization, shows the preparedness of PERTURBO for the future HPC architectures and for the Exascale computing. We also plan to make Perturbo available as a module on NERSC in the near future.



Download and Installation

Quantum Espresso, Wannier 90, and HDF5 Download and Installation

PERTURBO uses a few subroutines from the PWSCF and Phonon packages of Quantum Espresso (QE). Therefore, it needs to be compiled on top of QE. PERTURBO needs the output files from Wannier 90 (W90). In addition, PERTURBO uses the HDF5 format to store data. For the detailed instructions on the installation of these packages, please refer to their official websites: QE , W90 , HDF5 . If you run the calculations on a cluster or a supercomputer, these packages might be already pre-installed. Here we provide some brief instructions on the installation of these packages. Please note that these instructions can be different for your computing environement.

Quantum Espresso

> ℹ **Note:** The supported versions of QE is 7.0. For compatibility with the 6.4 and 6.5 QE versions, read here.

To download QE, one can use the `wget` command:

```
wget https://github.com/QEF/q-e/archive/qe-7.0.tar.gz
tar xvzf qe-7.0.tar.gz
cd q-e-qe-7.0
```

or to clone the package from the QE GitHub repository, specifying the version:

```
git clone https://github.com/QEF/q-e.git
cd q-e
git checkout qe-7.0
```

Once the package is downloaded run the configure command:

```
./configure
```

one can run `./configure --help` to get additional configure options (for the C, C++, Fortran compilers, etc.). Compile QE:

```
make pw ph pp
```

Wannier 90

We suggest to install Wannier 90 inside the QE folder.

Download the package with `wget`:

```
wget https://github.com/wannier-developers/wannier90/archive/v
3.0.0.tar.gz
tar xvzf v3.0.0.tar.gz
cd wannier90-3.0.0
```

or clone from the GitHub repository:

```
git clone https://github.com/wannier-developers/wannier90.git
cd wannier90
```

Compile Wannier 90:

```
cp ./config/make.inc.xxx ./make.inc
make
```

HDF5

To compile the HDF5 library, please download its source code from the official website . Once the source code is downloaded, please create an empty directory where the HDF5 library will be installed.

For example, the current directory is called *'mylib'*. We download the source code inside the directory called *'hdf5-1.12.0-source-codes'*. Now we are going to install the HDF5 library into a directory called *'hdf5-1.12.0'*. Create the empty directories:

```
mkdir hdf5-1.12.0
```

Generate a Makefile for compiling the HDF5 library using the fortran option: `--enable-fortran`. Please modify the `'prefix'` path to fit your case. Here we compile the serial HDF5 library:

```
cd hdf5-1.12.0-source-codes
./configure --prefix=mylib/hdf5-1.12.0 --enable-fortran
```

One can specify the compilers running the `./configure` command with the additional options: `CC=<c compiler>`, `CXX=<c++ compiler>`, `FC=<fortran compiler>`. For more information, run `./configure --help`. Compile HDF5:

```
make
make install
```

To check whether HDF5 was compiled correctly, one can run the test suite:

```
make test
```

Now we have the compiled HDF5 library inside the directory *'hdf5-1.12.0'*. Please use the directory path when compiling PERTURBO (`IFLAGS` and `HDF5_LIB` parameters in the PERTURBO *make.sys* file, see below).

> **Warning:** On some systems, one may need to disable the file locking by executing the following command:
> `export HDF5_USE_FILE_LOCKING=FALSE`

> **Note:** Quantum Espresso can be also compiled with the HDF5 library. If you would like to do so, please add the option `--with-hdf5=<your-hdf5-directory>` when configuring QE (more details here).

PERTURBO Download and Installation

Download

In order to get access to the code, please fill out this form.

To help us keep track of user number, we encourage each individual user to submit a separate request for code download. For example, research groups with multiple users should also have each user submit a request.

If the the form does not work, see here the intructions.

Clone from GitHub (or extract *.tar.gz*) into the QE directory. There are three subdirectories inside the directory *"perturbo"*:

- *"config"* contains the system-dependent makefiles *make.sys.XXX*
- *"pert-src"* contains the source code of `perturbo.x` to compute electron dynamics
- *"qe2pert-src"* contains the source code of the interface program `qe2pert.x`

The source code is supplemented by the tutorial examples input and output files. More details about the examples can be found in the Organization section.

Installation

There are two files in the *"perturbo"* directory, *Makefile* and *make.sys*. PERTURBO uses the config file *make.inc* of QE for most of the compiler options. The config file *make.sys* inside the directory *"perturbo"* specifies additional options required by PERTURBO. Modify *make.sys* to make it suitable for your system:

```
vim make.sys
```

specify the path to your HDF5 library:

```
IFLAGS += -L<path-to-hdf5-dir>/hdf5-1.12.0/include -lhdf5 -lhdf
5_fortran
HDF5_LIB = -L<path-to-hdf5-dir>/hdf5-1.12.0/lib -lhdf5 -lhdf5_f
ortran
```

Modify the two flags `FFLAGS` and `LDFLAGS` for your compilers, for example:

```
#for intel compiler
FFLAGS += -qopenmp -cpp
LDFLAGS += -qopenmp
```

or

```
#for gfortran compiler
FFLAGS += -fopenmp -x f95-cpp-input
LDFLAGS += -fopenmp
```

Once the file *make.sys* has been modified, you are ready to compile PERTURBO:

```
make
```

After the compilation, a directory called *'bin'* is generated, which contains two executables, `perturbo.x` and `qe2pert.x`.

Click here to see a video tutorial on this topic.

If the contact link does not work, in order to get access to the code, please write us an email to [perturbo AT caltech.edu] and provide the following information about you:

Name:
Organization:
Country:
I am going to use PERTURBO for:
GitHub username:

Bernardi Research Group

The PERTURBO code is developed in Marco Bernardi's research group at Caltech. For more information, your are invited to visit the group website .

Papers that used or cited Perturbo

1. T. Truttmann, J.-J. Zhou, I.-T. Lu, A. Rajapitamahuni, F. Liu, T. Mates, M. Bernardi, B. Jalan
   Combined experimental-theoretical study of electron mobility-limiting mechanisms in SrSnO3.
   Communications Physics **1**, 241, (2021)

2. H. Liu, I. Klein, J. Michelsen, S. Cushing
   Element-specific electronic and structural dynamics using transient XUV and soft X-ray spectroscopy.
   Chem **10**, 2569-2584, (2021)

3. I. Maliyov, J. Park, M. Bernardi
   Ab initio electron dynamics in high electric fields: Accurate prediction of velocity-field curves.
   Physical Review B **10**, L100303, (2021)

4. J.-J. Zhou, J. Park, I. Timrov, A. Floris, M. Cococcioni, N. Marzari, M. Bernardi
   Ab Initio Electron-Phonon Interactions in Correlated Electron Systems.
   Physical Review Letters **12**, 126404, (2021)

5. X. Qing, C. Zhang, J. Gong, S. Chen
   Ab initio study of photoelectric properties in ZnO transparent conductive oxide.
   Vacuum **191**, 110391, (2021)

6. T. Fan, A. Oganov
   AICON2: A program for calculating transport properties quickly and accurately.
   Computer Physics Communications **266**, 108027, (2021)

7. S. Mu, A. Rowberg, J. Leveillee, F. Giustino, C. Van de Walle
   First-principles study of electron transport in ScN.
   Physical Review B **7**, 075118, (2021)

8. R. Gupta, B. Dongre, J. Carrete, C. Bera
   Thermoelectric properties of the SnS monolayer: Fully ab initio and accelerated calculations.
   Journal of Applied Physics **5**, 054301, (2021)

9. B. Kozinsky, D. Singh
   Thermoelectrics by Computational Design: Progress and

Opportunities.
Annual Review of Materials Research **1**, 565-590, (2021)

10. M. Wais, K. Held, M. Battiato
Numerical solver for the time-dependent far-from-equilibrium Boltzmann equation.
Computer Physics Communications **264**, 107877, (2021)

11. N.-E. Lee, H.-Y. Chen, J.-J. Zhou, M. Bernardi
Facile ab initio approach for self-localized polarons from canonical transformations.
Physical Review Materials **6**, 063805, (2021)

12. S. Gao, H.-Y. Chen, M. Bernardi
Radiative properties of quantum emitters in boron nitride from excited state calculations and Bayesian analysis.
npj Computational Materials **1**, 85, (2021)

13. X. Tong, M. Bernardi
Toward precise simulations of the coupled ultrafast dynamics of electrons and atomic vibrations in materials.
Physical Review Research **2**, 023072, (2021)

14. A. Ganose, J. Park, A. Faghaninia, R. Woods-Robinson, K. Persson, A. Jain
Efficient calculation of carrier scattering rates from first principles.
Nature Communications **1**, 2222, (2021)

15. D. Desai, B. Zviazhynski, J.-J. Zhou, M. Bernardi
Magnetotransport in semiconductors and two-dimensional materials from first principles.
Physical Review B **16**, L161103, (2021)

16. A. Choi, P. Cheng, B. Hatanpää, A. Minnich
Electronic noise of warm electrons in semiconductors from first principles.
Physical Review Materials **4**, 044603, (2021)

17. Z. Gao, T. Zhu, K. Sun, J.-S. Wang
Highly Anisotropic Thermoelectric Properties of Two-Dimensional As2Te3.
ACS Applied Electronic Materials **4**, 1610-1620, (2021)

18. J. Lee, S. Zhang, D. Reichman
Constrained-path auxiliary-field quantum Monte Carlo for coupled electrons and phonons.
Physical Review B **11**, 115123, (2021)

19. A. Afzalian, E. Akhoundi, G. Gaddemane, R. Duflou, M. Houssa

Advanced DFT-NEGF Transport Techniques for Novel 2-D Material and Device Exploration Including HfS_2/WSe_2 van der Waals Heterojunction TFET and WTe_2/WS_2 Metal/Semiconductor Contact.
IEEE Transactions on Electron Devices 1-8, (2021)

20. A. Balvanz, J. Qu, S. Baranets, E. Ertekin, P. Gorai, S. Bobev
New n-Type Zintl Phases for Thermoelectrics: Discovery, Structural Characterization, and Band Engineering of the Compounds A2CdP2 (A = Sr, Ba, Eu).
Chemistry of Materials **24**, 10697-10707, (2020)

21. T. Sohier, M. Gibertini, N. Marzari
Profiling novel high-conductivity 2D semiconductors.
2D Materials (2020)

22. V. Jhalani, J.-J. Zhou, J. Park, C. Dreyer, M. Bernardi
Piezoelectric Electron-Phonon Interaction from Ab Initio Dynamical Quadrupoles: Impact on Charge Transport in Wurtzite GaN.
Physical Review Letters **13**, 136602, (2020)

23. J. Park, J.-J. Zhou, V. Jhalani, C. Dreyer, M. Bernardi
Long-range quadrupole electron-phonon interaction from first principles.
Physical Review B **12**, 125203, (2020)

24. H.-Y. Chen, D. Sangalli, M. Bernardi
Exciton-Phonon Interaction and Relaxation Times from First Principles.
Physical Review Letters **10**, 107401, (2020)

25. N.-E. Lee, J.-J. Zhou, H.-Y. Chen, M. Bernardi
Ab initio electron-two-phonon scattering in GaAs from next-to-leading order perturbation theory.
Nature Communications **1**, 1607, (2020)

26. I.-T. Lu, J. Park, J.-J. Zhou, M. Bernardi
Ab initio electron-defect interactions using Wannier functions.
npj Computational Materials **1**, 17, (2020)

27. J. Park, J.-J. Zhou, M. Bernardi
Spin-phonon relaxation times in centrosymmetric materials from first principles.
Physical Review B **4**, 045202, (2020)

28. J.-J. Zhou, M. Bernardi
Predicting charge transport in the presence of polarons: The beyond-quasiparticle regime in SrTiO3.
Physical Review Research **3**, 033138, (2019)

29. V. Jhalani, H.-Y. Chen, M. Palummo, M. Bernardi
Precise radiative lifetimes in bulk crystals from first principles: the case of wurtzite gallium nitride.
Journal of Physics: Condensed Matter **8**, 084001, (2019)

30. H.-Y. Chen, V. Jhalani, M. Palummo, M. Bernardi
Ab initio calculations of exciton radiative lifetimes in bulk crystals, nanostructures, and molecules.
Physical Review B **7**, 075135, (2019)

31. I.-T. Lu, J.-J. Zhou, M. Bernardi
Efficient ab initio calculations of electron-defect scattering and defect-limited carrier mobility.
Physical Review Materials **3**, 033804, (2019)

32. J.-J. Zhou, O. Hellman, M. Bernardi
Electron-Phonon Scattering in the Presence of Soft Modes and Electron Mobility in SrTiO3 Perovskite from First Principles.
Physical Review Letters **22**, 226603, (2018)

33. L. Agapito, M. Bernardi
Ab initio electron-phonon interactions using atomic orbital wave functions.
Physical Review B **23**, 235146, (2018)

34. H.-Y. Chen, M. Palummo, D. Sangalli, M. Bernardi
Theory and Ab Initio Computation of the Anisotropic Light Emission in Monolayer Transition Metal Dichalcogenides.
Nano Letters **6**, 3839-3843, (2018)

35. N.-E. Lee, J.-J. Zhou, L. Agapito, M. Bernardi
Charge transport in organic molecular semiconductors from first principles: The bandlike hole mobility in a naphthalene crystal.
Physical Review B **11**, 115203, (2018)

36. V. Jhalani, J.-J. Zhou, M. Bernardi
Ultrafast Hot Carrier Dynamics in GaN and Its Impact on the Efficiency Droop.
Nano Letters **8**, 5012-5019, (2017)

37. J.-J. Zhou, M. Bernardi
Ab initioelectron mobility and polar phonon scattering in GaAs.
Physical Review B **20**, 201201, (2016)

Perturbo Tutorials

In this section, we present the tutorials that describe all the capabilities of the PERTURBO code for a variety of materials. We focus on bulk silicon as a test case for the majority of calculation modes. When Si is not the best test case for a given tutorial, we will switch to another material.

Download tutorial files

We provide two repositories for the tutorials:

- perturbo-examples-light (~33 MB): contains the input files for the tutorials
- perturbo-examples-full (~24 GB): contains the input and output files for the tutorials.

We recommend users to download the input files from the *perturbo-examples-light* repository and then follow the steps of the tutorials. In case of a problem at some stage, the user can download the output from the *perturbo-examples-full* repository. Therefore, the large *perturbo-examples-full* repository is *only* for the occasional use.

In the tutorail text, we specify the path for a given tutorial section in the following way:

Directory: *example02-silicon-perturbo/perturbo/pert-bands/* link

The path is the same for the *-light* and *-full* repositories. We also provide a link to a given folder in the *-full* repository. The output files can be found in the *References* folder.

Organization of tutorials

In the tutorial, we first demostrate how to interface Quantum Espresso and Wannier90 with PERTURBO, in particular, how to generate the so-called '*prefix'_epwan.h5* HDF5 file (using the `qe2pert.x` executable), which will be later read by the main executable of PERTURBO (`perturbo.x`). This is done in the Quantum Espresso to Perturbo section and *example01-silicon-qe2pert* folder.

Next, we demonstrate the main PERTURBO capabilities:

- Interpolation Modes: intepolation of electronic bands, phonon dispersion, e-ph matrix elements
- Scattering Modes: imaginary part of e-ph self-energy and calculation of

the mean free path

- Transport Modes: electrical conductivity, mobility, and thermal transport
- Ultrafast Dynamics: real-time dynamics by time-stepping BTE

This is done in the *example02-silicon-perturbo* tutorial folder.

Finally, we provide the examples `qe2pert.x` step and and for some of the calculation modes of `perturbo.x` for other systems: Si with SOC, GaAs, graphene, and aluminum. This examples can be found in the *example03-…* – *example06-…* tutorial folders.

> ℹ **Note:** To run the main PERTURBO calculations (using `perturbo.x`), the `qe2pert.x` interface step must be accomplished. Every `perturbo.x` calculation mode relies on the '*prefix*'_epwan.h5 file. The `qe2pert.x` step can be avoided only if a user downloads the '*prefix*'_epwan.h5 file from the *example-…/qe2pert* folder.

Quantum Espresso to Perturbo

Before running electron dynamics calculations using `pertubo.x`, the user needs to carry out electronic and phonon calculations, with DFT and DFPT respectively. At present, Perturbo can read the output of DFT and DFPT calculations done with Quantum Espresso (QE). Once the relevant output files have been obtained from QE and Wannier90 (W90), the first step is to use `qe2pert.x` to compute the e-ph matrix elements on a coarse \(\mathbf{k}\) and \(\mathbf{q}\) point Brillouin zone grid, to obtain e-ph matrix elements in Wannier function basis, and to store the data into the HDF5 format for `perturbo.x` to read. The generation of this HDF5 file, called '*prefix*'_epwan.h5, is discussed in this section of the manual.

The preparation stage consists of five steps:

1. Run a self-consistent (scf) DFT calculation
2. Run a phonon calculation using DFPT
3. Run a non-scf (nscf) DFT calculation
4. Run Wannier90 to obtain Wannier functions
5. Run `qe2pert.x`

In the following, we use silicon as an example. The input files for QE and W90 are in the directory *"examples-perturbo/example02-silicon-qe2pert/pw-ph-wann"*. As a reference, we also provide the results in a directory called *"References"*.

Step 1: scf calculation

Directory: *example01-silicon-qe2pert/pw-ph-wann/scf/* link

Run an SCF calculation and obtain the QE *'prefix'.save* directory. In this case, we obtain *./tmp/si.save*, which is needed for phonon and nscf calculations.

Step 2: phonon calculation

Directory: *example01-silicon-qe2pert/pw-ph-wann/phonon/* link

We provide an example input file *ph-ref.in* for phonon calculations in QE, and two shell scripts (`ph-submit.sh` and `ph-collect.sh`) to set up and run a separate phonon calculation for each $\mathbf{q}$ point and collect the results. The user can modify the reference input file and the two shell scripts to use them for their material of choice and on their computing system.

> **Note:** The provided scripts are created for the PBS scheduler, for the SLURM scheduler, please, refer to the SLURM_scipts folder in the *phonon* directory.

In this step, make sure that the number of $\mathbf{q}$ points is commensurate with the number of $\mathbf{k}$ points used in the nscf and Wannierization calculations. For example, a $\mathbf{q}$ grid of 8x8x8 can be used with a wannierization $\mathbf{k}$ grid of 8x8x8 or 16x16x16, but not with a 10x10x10 or 12x12x12 grid.

Remember to copy the QE *'prefix'.save* directory from the scf run to the current directory:

```
$ cp -r ../scf/tmp ./
```

To obtain the number of irreducible $\mathbf{q}$ points in the phonon calculation, edit the *ph-submit* file and set `mode='gamma'` to run a Gamma-point phonon calculation.

```
$ vim ph-submit.sh
......
set mode='gamma'
......
$ ./ph-submit
```

The shell script creates a directory called *ph-1*. Change into that directory and open the file *ph.out* to read the number of irreducible $\mathbf{q}$ points in the phonon calculation.

```
$ cd ph-1
$ vi ph.out
```

In our silicon example, the total number of $\mathbf{q}$ points is 29. It is fine to forgo the previous step and obtain the number of $\mathbf{q}$ points some other way. Once this information is available, open again the shell script *ph-submit*. Change the starting number from 1 to 2 and the final number to the total number of irreducible $\mathbf{q}$ points.

```
$ vi ph-submit.sh
......
change (NQ=1; NQ<=8; NQ++) to (NQ=2; NQ<=29; NQ++)
......
$ ./ph-submit
```

The shell script creates one directory (*ph-#*) for each $\mathbf{q}$ point. Once the calculations are done, we collect all the phonon data into a directory called *save*, created by running the shell script *ph-collect.sh*.

```
$ ./ph-collect.sh
```

The *save* directory contains all the information needed for PETURBO to interface with QE. These include the dynamical matrix files, phonon perturbation potentials, and the patterns.

The reference input file and scripts can be modified to run calculations for different materials. We recommend the user to become familiar with phonon calculations in QE to perform this step.

Since phonon calculations using DFPT can be computationally expensive, it is often useful to estimate the number of irreducible $\mathbf{q}$ points before running the phonon calculation. Note however that this step is optional.

Step 3: nscf calculation

Directory: *example01-silicon-qe2pert/pw-ph-wann/nscf/* link

We now run the nscf calculations needed to generate the wavefunctions on the full $\mathbf{k}$ point grid, which we'll need both for generating Wannier functions with Wannier90 and for forming the coarse-grid e-ph matrix elements in Perturbo. Make sure that the number of k points is commensurate with the number of $\mathbf{q}$ points used for phonons, otherwise, qe2pert.x will stop. Remember to copy the QE *'prefix'.save* directory from the scf calculation the current directory:

```
$ cp -r ../scf/tmp ./
```

Then run the nscf calculation with QE.

Step 4: Wannier90 calculation

Directory: *example01-silicon-qe2pert/pw-ph-wann/wann/* link

Note: Requires Wannier90 v3.0.0 and higher.

The directory contains two input files, one for `wannier.x` and the other for `pw2wannier90.x`. In the input file *si.win*, we instruct Wannier90 to write two important quantities for `qe2pert.x`, the $U(\mathbf{k})$, $U^{\text{dis}}(\mathbf{k})$ matrices and the position of the Wannier function centers, using: `write_u_matrices=true` and `write_xyz=true`.

We create tmp directory:

```
$ mkdir tmp
```

and change into it. We soft link to the QE *'prefix'.save* directory obtained in the nscf calculation:

```
$ cd tmp
$ ln -sf ../../nscf/tmp/si.save
```

We then run Wannier90. The important output files for `qe2pert.x` are *si_u.mat*, *si_u_dis.mat*, and *si_centres.xyz*. For disentangled bands, there would be no *'prefix'_u_dis.mat*. We encourage the user to become familiar with Wannier90 to run this step for different materials.

The user has to run Wannier90 3.0 or higher, since otherwise the U matrices cannot be printed out.

Step 5: Running qe2pert.x

Directory: *example01-silicon-qe2pert/qe2pert/* link

We are ready to compute the e-ph matrix elements on the coarse \(\mathbf{k}\) point (determined by the nscf step) and \(\mathbf{q}\) point (determined by the phonon step) Brillouin zone grids. First, copy or link the electronic and phonon calculation results to the current directory.

```
$ cd qe2pert
$ mkdir tmp
$ cd tmp

$ #link to the nscf .save directory
$ ln -sf ../../pw-ph-wann/nscf/tmp/si.save
$ cd ../

$ #link to the wannier information
$ ln -sf ../wann/si_u.mat
$ ln -sf ../wann/si_u_dis.mat
$ ln -sf ../wann/si_centres.xyz
```

Here we show the input file (*qe2pert.in*) for the executable `qe2pert.x`:

```
&qe2pert
 prefix='si'
 outdir='./tmp'
 phdir='../pw-ph-wann/phonon/save'
 nk1=8, nk2=8, nk3=8
 dft_band_min = 1
 dft_band_max = 16
 num_wann = 8
 lwannier=.true.
 load_ephmat = .false.
 system_2d = .false.
/
```

The description of the input parameters:

- prefix: needs to be the same as the prefix used in the input files for QE.
- outdir: contains the save directroy obtained from the nscf calculations. The calculated e-ph matrix elements will be stored in this directory.
- phdir: is the save directory inside which we collected all the phonon information.
- nk1, nk2, nk3: are the number of \(\mathbf{k}\) points along each direction used in the nscf and Wannier90 calculations.
- dft_band_min and dft_band_max: determine the range of bands we are interested in, and should be the same as the values used in the Wannierization process. For example, if we used 40 bands in the nscf calculation and we excluded bands 1-4 and 31-40 in the Wannierization, then `dft_band_min=5` and `dft_band_max=30`.
- num_wann: the number of Wannier functions.
- lwannier: a logical flag. When it is `.true.`, the e-ph matrix elements are computed using the Bloch wave functions rotated with the Wannier unitary matrix; if `.false.`, the e-ph matrix elements are computed using the Bloch wave functions, and the e-ph matrix elements are then rotated using the Wannier unitary matrix. By default, it is `.true.` to reduce computational cost.
- load_ephmat: a logical flag. If `.true.`, reuse e-ph matrix elements in Bloch function basis computed previously. This is useful if you want to test different Wannier bases. For example, you could first run `qe2pert.x` with `lwannier=.false.`, and then rerun `qe2pert.x` with `lwannier=.false.` and `load_ephmat=.true.` with different Wannier unitary matrix.
- system_2d: if the materials is two-dimensional, so that in one direction only one \(\mathbf{k}\) point is used, set it to `.true.`; the default is `.false.`.

Now we are ready to run the e-ph matrix elements:

```
export OMP_NUM_THREADS=4
$ mpirun -n 2 qe2pert.x -npools 2 -i qe2pert.in > qe2pert.out
```

This task is usually time-consuming on a single core, but it can be made much faster (minutes) on multiple cores. The executables `qe2pert.x` employ hybrid parallelization (MPI plus OpenMP), e.g. 2 MPI processes and each process span 4 OpenMP threads in this example.

Note: The number of pools (-npools) has to be equal to the number of MPI processes (-np or -n), otherwise the code will stop.

To speed up the calculations, the users could increase the number of OpenMP threads and MPI processes. Threads with OpenMP are particularly useful when the RAM (memory) of computing nodes is limited. The memory consumption reduces to minimum when using 1 MPI process per node and setting `OMP_NUM_THREADS` to the number of cores per node.

Once the calcalculation has completed, we obtain the output file *si_epwan.h5*, which is an HDF5 database with all the information needed to run `perturbo.x` (which is described in the next section).

Click here to see a video tutorial on this topic.

Structure of epwan.h5 HDF5 file

The HDF5 file *'prefix'_epwan.h5* has the following data structure:

basic_data

Variables	Data type	Meaning
alat	real	lattice constant in atomic unit (Bohr)
at	real; dimension (1:3,1:3)	lattice vectors in unit of \(a\), where \(a\) is the lattice constant
bg	real; dimension (1:3,1:3)	reciprocal lattice vectors in unit of \(2\pi/a\), where \(a\) is the lattice constant
nat	intger	number of atoms in the unit cell
tau	real; dimension (1:3,1:3)	atomic positions in *a* units, where \(a\) is the lattice constant

Variables	Data type	Meaning
volume	real	volume for the unit cell in unit of Bohr\(^3\)
nsym	integer	number of symmetry operations for the simulation cell
symop	integer; dimension(1:3,1:3,1:nsym)	symmetry operations
kc_dim	integer; dimension (1:3)	number of coarse \(\mathbf{k}_c\)-points in each direction for the electronic system
spinor	integer	if spinor==1, the electron is spinor; if spinor==0, nonspinor
polar_alpha	real	broadening of the Lorezian function for generating the weighting for polar electron-phonon matrix element calculations
epsil	real; dimesion (1:3,1:3)	the dielectric tensor
qc_dim	integer; dimension (1:3)	number of coarse **q**\(_c\)-points in each direction for the phononic system
mass	real; dimension (1:nat)	atomic mass in atomic unit
zstar	real; dimension (1:3,1:3,1:nat)	the effective charge for each atom
system_2d	Integer	if system_2d==1, the system is a 2D material; if system_2d==0, not a 2D material

Variables	Data type	Meaning
num_wann	integer	number of Wannier functions used in the Wannierization
wannier_center	real; dimesion (1:3,1:num_wan)	the center of Wannier functions in Cartesian unit
wannier_center_cryst	real; dimesion (1:3, 1:num_wan)	the center of Wannier functions in crystal unit

electronic_wannier

Variables	Datatype	Meaning
hopping_r*x*	real; dimension(1:nr_el) where nr_el is the number of lattice points in the electronic system, and note that nr_el depends on which Wannier function is refered to	real part of the electronic Hamiltonian in Wannier basis; *x* is the index for the lattice points in the Wigner-Seitz cell
hopping_i*x*	real; dimension(1:nr_el) where nr_el is the number of lattice points in the electronic system, and note that nr_el depends on which Wannier function is refered to	Imaginary part of the electronic Hamiltonian in Wannier basis; *x* is the index for the lattice points in the Wigner-Seitz cell

eph_matrix_wannier

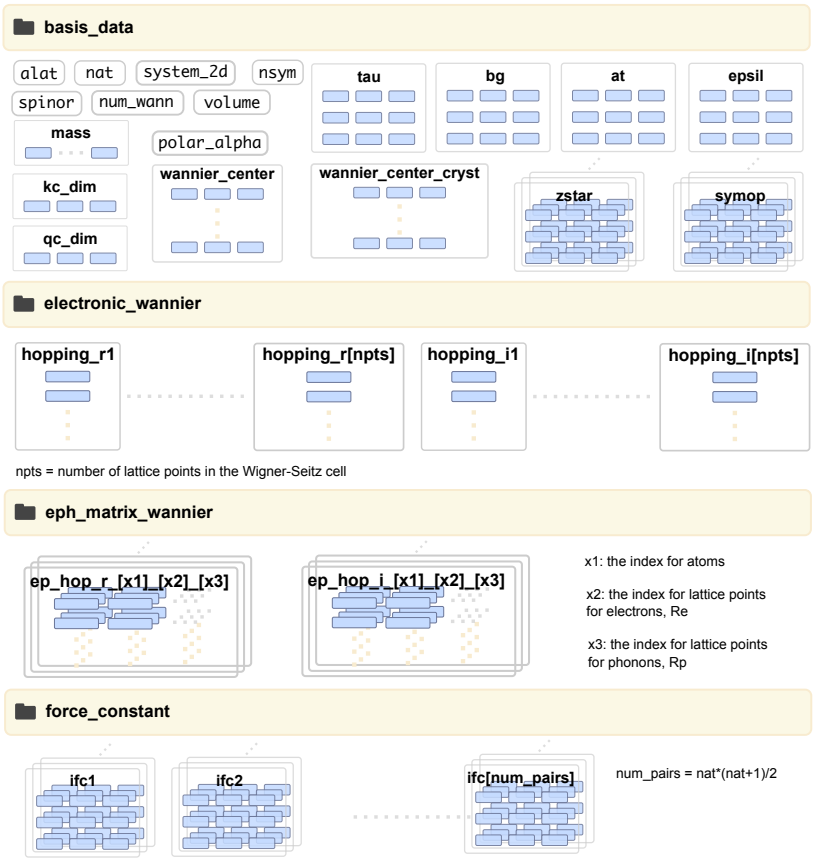
Variables	Datatype	Meaning
ep_hop_r_*x*_*x*_*x*	real; dimension (1:nat,1:nr_el,1:nr_ph) where nat is the number of atoms, nr_el is the number of lattice points in the electronic system, and nr_el is the number of lattice points in the phononic system	real part of the electron-phonon matrix elements in Wannier basis; 1st *x*: index for the atoms; 2nd *x*: index for R for electrons; 3rd *x*: index for R for phonons

Variables	Datatype	Meaning
ep_hop_i_*x*_*x*_*x*	real; dimension (1:nat,1:nr_el,1:nr_ph) where nat is the number of atoms, nr_el is the number of lattice points in the electronic system, and nr_el is the number of lattice points in the phononic system	imaginary part of the electron-phonon matrix elements in Wannier basis; 1st *x*: index for the atoms; 2nd *x*: index for **R** for electrons; 3rd *x*: index for **R** for phonons

force_constant

Variables	Datatype	Meaning
Ifc*x*	complex; dimesion (1:3,1:3,1:nr_ph) where nr_ph: number of lattice points for the phononic system	force constant

The ‘prefix’_epwan.h5 file structure can be schematically represented as follows:



Running Perturbo

In the following, we use silicon as an example to demonstrate the features of PERTURBO (see the directory *"example02-silicon-perturbo/perturbo"*). **To run** `perturbo.x` **one first needs to generate the file *'perfix'_epwan.h5*** (in this case, *si_epwan.h5*), which is prepared using `qe2pert.x` as we discuss in this section. The file *si_epwan.h5* is inside the directory *"example02-silicon-perturbo/qe2pert"* . We also provide reference output results in the directory *"References"*.

Calculation modes

Each PERTURBO feature corresponds to a separate run of `perturbo.x`, called a *calculation mode*. For each calculation mode, at least two files must be always presented in the run folder: 1) *'perfix'_epwan.h5* and 2) input file (usually called *pert.in*). Different calculation modes require some additional input files. The parameters specified in the input file *pert.in* depend on the calculation mode. The main input parameter controlling the calculation mode is `calc_mode`. Here is a brief summary of the `calc_mode` options, and the corresponding tasks carried out by PERTURBO:

- `'bands'`: interpolate electronic band structures using Wannier functions.
- `'phdisp'`: interpolate phonon dispersion by Fourier transforming real-space interatomic force constants.
- `'ephmat'`: interpolate e-ph matrix elements using Wannier functions.
- `'setup'`: setup for transport calculations or carrier dynamics simulations.
- `'imsigma'`: compute the e-ph self-energy for electronic crystal momenta read from a list.
- `'meanfp'`: compute the e-ph mean free path, also output the corresponding band velocity and relaxation time.
- `'trans'`: compute electrical conductivity for metals, semiconductors, and insulators, or carrier mobility for semiconductors, using either the state-dependent RTA approach or the iterative approach of the BTE.
- `'trans-pp'`: postprocessing of the 'trans' calculation, compute the Seebeck coefficient.
- `'dynamics-run'`: ultrafast hot carrier dynamics via the time-dependent Boltzmann transport equation.
- `'dynamics-pp'`: postprocessing of the 'dynamics-run' calculation,

compute the BZ-averaged energy-dependent carrier population.

The next sections of the tutorial are dedicated to the detailed description of each calculation mode. For a brief overview, a list of the required input files, and an example input file for each calculation mode, please refer to the Interactive workflow page.

Interpolation of Bands, Phonon Dispersion, and e-ph Matrix Elements

In this page, we will discuss the calculation modes of PERTURBO related to the interpolation. Here are the value of `calc_mode` for the interpolation modes:

- `'bands'`: interpolate electronic band structures using Wannier functions.
- `'phdisp'`: interpolate phonon dispersion by Fourier transforming real-space interatomic force constants.
- `'ephmat'`: interpolate e-ph matrix elements using Wannier functions.

Electronic bands

calc_mode = 'bands'

Directory: *example02-silicon-perturbo/perturbo/pert-bands/* link

Computes: Interpolated electronic band structure given an electronic crystal momentum path.

Users specify three variables in the input file (*pert.in*)

- prefix: the same prefix used in **'prefix'_epwan.h5**
- calc_mode: set to `'bands'`
- fklist: the filename of a file containing the high-symmetry crystal momentum path or k list

Here is the input file or namelist (*pert.in*):

```
&perturbo
 prefix = 'si'
 calc_mode = 'bands'
 fklist = 'si_band.kpt'
/
```

In this example, `fklist='si_band.kpt'`, the file *si_band.kpt* containing the \(\mathbf{k}\) point list:

```
6
0.500   0.500   0.500   50
0.000   0.000   0.000   50
0.500   0.000   0.500   20
0.500   0.250   0.750   20
0.375   0.375   0.750   50
0.000   0.000   0.000   1
```

The first line specifies how many lines there are below the first line. Columns 1-3 give, respectively, the \(x\), \(y\), and \(z\) coordinates of a crystal momentum **in crystal coordinates**. The last column is the number of points from the current crystal momentum to the next crystal momentum. One can also provide an explicit \(\mathbf{k}\) point list, rather than specifying the path, by providing the number of \(\mathbf{k}\) points in the first line, the coordinates of each \(\mathbf{k}\) point, and setting the values in the last column to 1.

Before running `perturbo.x`, remember to put *si_epwan.h5* in the current directory "pert-band" since `perturbo.x` needs to read *si_epwan.h5*. You may choose to copy the HDF5 file using

`$ cp ../../qe2pert/si_epwan.h5 .`

But the size of the HDF5 file is usually quite large, creating a soft link that point to the original HDF5 file is strongly recommended:

`$ ln -sf ../../qe2pert/si_epwan.h5`

Run `perturbo.x`:

```
$ mpirun -n 1 perturbo.x -npools 1 -i pert.in > pert.out
```

> ℹ **Note:** The number of pools (-npools) has to be equal to the number of MPI processes (-np or -n), otherwise the code will stop.

It takes just a few seconds to obtain the interpolated band structure. We obtain an output file called *'prefix'.bands* (in this case, *si.bands*) with the following format:

```
0.0000000      0.50000    0.50000    0.50000     -3.4658249872
......
3.7802390      0.00000    0.00000    0.00000     -5.8116812661

......
......

0.0000000      0.50000    0.50000    0.50000     13.6984850767
......
3.7802390      0.00000    0.00000    0.00000      9.4608102223
```

Note that there are 8 blocks in this example, one for each of the 8 bands, because we use 8 Wannier functions in the Wannierization procedure in this example. The 1st column is an irrelevant coordinate used to plot the band structure. The 2nd to 4th columns are the \(x\), \(y\), and \(z\) coordinates of the crystal momenta **in crystal coordinates**. The 5th column is the energy, in eV units, of each electronic state.

Phonon dispersion
calc_mode = 'phdisp'

Directory: *example02-silicon-perturbo/perturbo/pert-phdisp/* link

Computes: Interpolated phonon dispersions along a given crystal momentum path.

Users specify three variables in the input file (*pert.in*):

- prefix: the same prefix used in **'prefix'_epwan.h5**
- calc_mode: set to 'phdisp'
- fqlist: the filename of a file containing the high-symmetry crystal momentum path or q list

Here is the input file (*pert.in*):

```
&perturbo
 prefix = 'si'
 calc_mode = 'phdisp'
 fqlist = 'si_phdisp.qpt'
/
```

In this example, `fqlist='si_phdisp.qpt'`, and the file *si_phdisp.qpt* contains a crystal momentum path or list with the same format as the file specified in `fklist` (in the previous section).

Remember to link (or copy) *si_epwan.h5* in the current directory using

`ln -sf ../../qe2pert/si_epwan.h5`.

Run `perturbo.x`:

```
$ mpirun -n 1 perturbo.x -npools 1 -i pert.in > pert.out
```

It takes a few seconds to obtain the phonon dispersion. We obtain an output file called *'prefix'.phdisp* (in this case, *si.phdisp*) with the following format:

```
0.0000000      0.50000    0.50000    0.50000     12.9198400723
......
3.7802390      0.00000    0.00000    0.00000     -0.0000024786

......
......

0.0000000      0.50000    0.50000    0.50000     45.6922098051
......
3.7802390      0.00000    0.00000    0.00000      0.0000014170
```

Note that there are 6 blocks, one for each of the to 6 phonon modes in silicon. The 1st column an irrelevant coordinate used to plot the phonon dispersion. The 2nd to 4th columns are the \(x\), \(y\), and \(z\) coordinates of the crystal momenta, in crystal coordinate. The 5th column is the phonon energy in meV units.

E-ph matrix elements
calc_mode = 'ephmat'

🗀 **Directory:** *example02-silicon-perturbo/perturbo/pert-ephmat/* link

Computes: The absolute values of the e-ph matrix elements, summed over the number of electronic bands, given two lists of $\mathbf{k}$ and $\mathbf{q}$ points. In a typical scenario, one computes the e-ph matrix elements for a chosen $\mathbf{k}$ point as a function of $\mathbf{q}$ point.

Requires to specify at least 7 variables:

- prefix: the same prefix as in *'prefix'_epwan.h5*
- calc_mode: set to 'ephmat'
- fklist: the file containing a list of $\mathbf{k}$ points (for the format of the list, please see the section on `calc_mode='bands'`)
- fqlist: the file containing a list of $\mathbf{q}$ points (for the format of the list, please see the section on `calc_mode='bands'`)
- band_min, band_max: bands used for the band summation in computing e-ph matrix elements
- phfreq_cutoff: phonon energy (meV) smaller than the cutoff will be ignored

In a typical scenario, the user wants to check if the interpolated e-ph matrix elements match with the density functional perturbation theory (DFPT) result. **Here we assume that users know how to obtain the DFPT e-ph matrix elements from the PHONON package in QE.**

Here is the input file (*pert.in*):

```
&perturbo
 prefix = 'si'
 calc_mode = 'ephmat'
 fklist = 'eph.kpt'
 fqlist = 'eph.qpt'

 band_min = 2
 band_max = 4

 phfreq_cutoff = 1   !meV
/
```

In this example, we compute the e-ph matrix elements summed over the bands from 2 to 4. The band index here refers to the band index of the Wannier functions, and it may not be the same as the band index in the DFT output from QE because sometimes bands are excluded in the Wannierization procedure. Make sure you know band range appropriate for your calculation, and provide accordingly band_min and band_max.

The variable phfreq_cutoff is used to avoid numerical instabilities in the phonon calculations, and we recommend using a value between 0.5 and 2 meV (unless you know that phonons in that energy range play a critical role). Do not set phfreq_cutoff to a large value, otherwise too many phonon modes will be excluded from the calculations.

For the format of fklist or fqlist files, please refer to the section on calc_mode='bands'.

Before running `perturbo.x`, ensure that three files exist in the current directory *"pert-ephmat"*:

- *'prefix'_epwan.h5*: here *si_epwan.h5*
- *fklist*: here *eph.kpt*
- *fqlist*: here *eph.qpt*

Run `perturbo.x`:

```
$ mpirun -n 1 perturbo.x -npools 1 -i pert.in > pert.out
```

The calculation typically takes a few minutes. The output file, called *'prefix'.ephmat*, contains the absolute values of the e-ph matrix elements summed over bands from band_min to band_max. In our example, we obtain the output file *si.ephmat*, which is shown next:

```
# ik    xk     iq     xq     imod   omega(meV)    deform. po
t.(eV/A)     |g|(meV)
  1   0.00000   1   0.00000  001    12.919840     0.21992730838
2E+00   0.118026594146E+02
  ......
  ......
```

The 1st column is a dummy index for the \(\mathbf{k}\) point. The 2nd column is the \(\mathbf{k}\) point coordinate used for plotting. The 3rd and 4th columns are the dummy index and the \(\mathbf{q}\) point coordinate used for plotting, respectively. The 5th column is the phonon mode index. The 6th column is the phonon energy (in meV). The 7th column is the deformation potential (in eV/Å units), namely the expectation value of the phonon perturbation potential with respect to the initial and final electronic states. The 8th column is the absolute values of the e-ph matrix elements (meV units) summed over the number of bands specified by the user.

Electron-phonon Scattering

The following calculation modes (`calc_mode` parameter) are related to the e-ph scattering computation:

- `'setup'`: setup for transport calculations or carrier dynamics simulations.
- `'imsigma'`: compute the e-ph self-energy for electronic crystal momenta read from a list.
- `'meanfp'`: compute the e-ph mean free path, also output the corresponding band velocity and relaxation time.

Setup electron k- and phonon q-grids *calc_mode = 'setup'*

Directory: *example02-silicon-perturbo/perturbo/pert-setup-electron/* link

Computes: Set up transport property calculations by providing \(\mathbf{k}\) points, \(\mathbf{k}\) point tetrahedra and (if needed) finding chemical potentials for given carrier concentrations. Also computes the density of states.

Requires to specify up to 14 variables in the input file (*pert.in*)

- prefix: same prefix as in *'prefix'_epwan.h5*
- calc_mode: set to 'setup'
- hole: By default, `hole` is set to `.false.`. Set it to .true. **only** when computing hole mobility of a semiconductor. if *hole* is .true., `perturbo.x` computes hole concentration, instead of electron concentration.
- boltz_kdim: number of \(\mathbf{k}\) points along each dimension of a \(\mathbf{k}\) point grid for the electrons momentum. This Gamma-centered Monkhorst-Pack \(\mathbf{k}\) point grid is employed to compute the mobility or conductivity.
- boltz_qdim: number of \(\mathbf{q}\) points along each dimension of a uniform grid for the phonon momentum; the default is that `boltz_qdim(i)=boltz_kdim(i)`. If users need the size as same as the \(\mathbf{k}\) grid, no need to specify these variables. Only phonons with mmentum on the \(\mathbf{q}\) grid are considered in the calculations of

e-ph scattering.

- boltz_emin, boltz_emax: energy window (in eV units) used to compute transport properties. The suggested values are from 6 k_BT below E_F (boltz_emin) to 6k_BT above E_F (boltz_emax), where E_F is the Fermi energy, k_B the Boltzmann constant, and T is temperature in K units.
- band_min, band_max: band window for transport property calculations
- ftemper: the filename of a file containing the temperature(s), chemical potential(s), and corresponding carrier concentration(s) for transport property calculations. Either chemical potentials or carrier concentrations is required dependending on the calculation setting.

Here is the input file (*pert.in*):

```
&perturbo
 prefix      = 'si'
 calc_mode   = 'setup'

 boltz_kdim(1) = 80
 boltz_kdim(2) = 80
 boltz_kdim(3) = 80

 boltz_emin = 6.4
 boltz_emax = 6.9
 band_min = 5
 band_max = 6

 ftemper  = 'si.temper'
/
```

In the input file *pert.in*, we use a \(\mathbf{k}\) grid of 80 x 80 x 80 for electrons, which corresponds to boltz_kdim(i)=80, and use a \(\mathbf{q}\) grid for phonons of the same dimension as the \(\mathbf{k}\) grid. When a phonon \(\mathbf{q}\) grid different from the electron \(\mathbf{k}\) grid is desired, the user need to provide the \(\mathbf{q}\) grid variables *boltz_qdim(1)*, *boltz_qdim(2)*, and *boltz_qdim(3)* in the input file.

In this example, we want to compute the mobility of the electron carrier, so we choose an energy window that includes the conduction band minimum. Here the energy window is between 6.4 (*boltz_emin*) and 6.9 eV (*boltz_emax*), and the conduction band minimum is at 6.63 eV in this case. We include the two lowest conduction bands, with band indices 5 and 6 (*band_min* and *band_max*).

The 'setup' calculation finds all the relevant $\mathbf{k}$ points (both irreducible and reducible $\mathbf{k}$ points) and the tetrahedron needed for BZ integration for the given energy window and band window computes the DOS at the given energy window. It also computes carrier concentrations at given chemical potentials or determines the chemical potentials that corresponding to the given carrier concentrations, depending on the setting in the ftemper file.

In this case, the ftemper file `si.temper` has the following format:

```
1 T
300.00   6.52   1.0E+18
```

The integer in the first line is the number of (temperature, chemical potential) settings at which we want to perform the transport calculations. Each of the following lines contains three values, the temperature (K), Fermi level (eV), and carrier concentration (cm^{-3} in 3D materials or cm^{-2} in 2D materials).

The logical variable in the first line indicates whether to compute the carrier concentration for the input chemical potential (if `F`) or determine the chemical potential corresponding to the input carrier concentration (if `T`), thus only one of the chemical potential column and carrier concentration column in the *ftemper* file is meaningful.

The logical variable is only used in the `'setup'` calculation. In all the other calc_mode options, `perturbo.x` reads the chemical potential column and ignores the carrier concentration column (and the logical variable). If one wants to perform transport calculations at given carrier concentrations, then set the logical variable to `T` in `'setup'` calculations. `perturbo.x` will find the corresponding chemical potentials and update the *ftemper* file accordingly (overwrite the chemical potential and carrier concentration columns and set the logical variable to `F`).

> **Note:** `perturbo.x` only search for chemical potentials within the given energy window, try extending the energy window if the updated *ftemper* file does not show reasonable carrier concentrations.

Run `perturbo.x` with the following command (remember to link or copy *'prefix'_epwan.h5* in the current directory):

```
$ mpirun -n 1 perturbo.x -npools 1 -i pert.in > pert.out
```

The calculation will take a few minutes or longer, depending the number of $\mathbf{k}$ and $\mathbf{q}$ points and the size of the energy window. We obtain 4 output files (*'prefix'.doping*, *'prefix'_tet.h5*, *'prefix'_tet.kpt*, and *'prefix'.dos*):

- *'prefix'.doping* contains chemical potentials and carrier concentrations for each tempearture of interest. The format is easy to understand so we do not show it here. Please take a look at the file by yourself.
- *'prefix'_tet.h5* contains information on the $\mathbf{k}$ points (both in the irreducible wedge and full grid) and the associated $\mathbf{k}$ point tetrahedra in the energy window of interest. This file will be used to compute transport properties. Users familiar with HDF5 can read and manipulate this file with the standard HDF5 commands. The other users can just ignore the data stored in the file.
- *'prefix'_tet.kpt* contains the coordinates (in crystal units) of the irreducible $\mathbf{k}$ points in the energy window of interest. Note that the irreducible $\mathbf{k}$ points coordinates is already included in *'prefix'_tet.h5*, we output to this file in a format compatiable with that of *fklist* discussed in the calculation mode `'bands'` (above) or `'imsigma'` (below).
- *'prefix'.dos* contains the density of states (number of states per eV per unit cell) as a function of energy (eV). The format is easy to understand so we do not show it here. The density of states sets the phase space for several electron scattering processes, so it is convenient to compute it and print it out.

In our example, since we used `'T'` in the first line of ftemper, a new *ftemper* file is generated as output: that the *ftemper* file *'si.temper'* has now become:

```
1 F
300.00   6.5504824219   0.9945847E+18
```

Note how `perturbo.x` has computed the chemical potential (second entry in the second row) for the given temperature and carrier concentration (first and third entries of the second row). The logical variable in the first line is now `'F'`, and *si.temper* can now be used as is in subsequent calculations.

The above explanation focuses on electrons. For holes carriers, please refer to *"example02-silicon-perturbo/perturbo/pert-setup-hole"*, link . In the input file for holes, remember to use `hole=.true.` (default: `hole=.false.`), and choose an appropriate energy window and the band indices for holes.

Imaginary part of e-ph self-energy
calc_mode = 'imsigma'

Directory: *example02-silicon-perturbo/perturbo/pert-imsigma-electron/*
link

Computes:
The imaginary part of the lowest-order (so-called 'Fan') e-ph self-energy, $\operatorname{Im}\Sigma$, for states in a range of bands and with crystal momenta $\mathbf{k}$ read from a list (this list can be obtained from `calc_mode='setup'` or created manually). The scattering rates can also be obtained using $2 \operatorname{Im}\Sigma /\hbar$.

Variables in the input file (*pert.in*)

- prefix: the same prefix as the file *'prefix'_epwan.h5*
- calc_mode: set to `'imsigma'`
- band_min, band_max: bands used for transport property calculations
- ftemper: the filename of a file containing temperature, chemical potential, and carrier concentration values (see the format)
- fklist: the filename of a file containing the coordinates of a given electron $\mathbf{k}$ point list (see the format)
- phfreq_cutoff: the cutoff energy for the phonons. Phonon with their energy smaller than the cutoff (in meV) is ignored; 0.5-2 meV is recommended.
- delta_smear: the broadening (in meV) used for the Gaussian function used to model the Dirac delta function
- fqlist: the filename of a file containing the coordinates of a given phonon $\mathbf{q}$ point list will be used to compute the e-ph self-energy. For the format, see the section on the calculation mode `'bands'`. This is optional. If `fqlist` is absent or `fqlist_=''`, random $\mathbf{q}$ points will be generated (see below).
- sampling: sampling method for random $\mathbf{q}$ points used in e-ph self-energy calculation. The default value is `'uniform'`, indicates sampling random $\mathbf{q}$ points in the first BZ following uniform

distribution. Another option is `'cauchy'`, sampling random $\mathbf{q}$ points following Cauchy distribution, which is useful for polar materials. Note that random $\mathbf{q}$ points from other importance sampling methods or $\mathbf{q}$ points on regular MP grid is also possible, one just needs to pre-generate the $\mathbf{q}$ points list to a file, and pass the file to `perturbo.x` via `fqlist`.

- cauchy_scale: the width of the Cauchy function; used only when sampling is 'cauchy'.
- nsamples: number of random $\mathbf{q}$ points sampled to compute the imaginary part of the e-ph self-energy for each $\mathbf{k}$ point

Here is the input file (*pert.in*):

```
&perturbo
 prefix      = 'si'
 calc_mode   = 'imsigma'

 fklist   = 'si_tet.kpt'
 ftemper  = 'si.temper'

 band_min = 5
 band_max = 6

 phfreq_cutoff = 1 ! meV
 delta_smear = 10 ! meV

 sampling = 'uniform'
 nsamples = 1000000
/
```

In the current example, we compute the imaginary part of the e-ph self-energy of $\mathbf{k}$ points in the *fklist* file (in this case, we use the irreducible Monkhorst-Pack $\mathbf{k}$ point list in *si_tet.kpt* obtained from the calculation mode `'setup'`). Note that if one is only interested in a high symmetry line, one can provide $\mathbf{k}$ point path in the *fklist* file instead. The temperature, chemical potential for computing the e-ph self-energy are given in the *ftemper* file, *si.temper*, obtained from the perturbo `'setup'` process (the carrier concentration column is ignored in `'imsigma'` calculation). Note that `perturbo.x` will do calculations, at once, for as many combinations of temperature and chemical potential as are specified in the lines below the first of *ftemper*.

Here we use a uniform random sampling (`sampling='uniform'`) with 1 million random \(\mathbf{q}\) points (`nsample=1000000`). The phonon frequency cutoff is 1 meV (`phfreq_cutoff=1`), and the smearing for the Gaussian function is 10 meV (`delta_smear=10`).

Before running `perturbo.x` , remember to link or copy *'prefix'_epwan.h5* in the current directory.

```
export OMP_NUM_THREADS=4
$ mpirun -n 8 perturbo.x -npools 8 -i pert.in > pert.out
```

This task is usually time-consuming on a single core, thus we run this calculation on multiple cores (32 cores in this case) using hybrid MPI plus openMP parallelization.

We obtain two output files:

- *'prefix'.imsigma* contains the computed imaginary part of the e-ph self-energy
- *'prefix'.imsigma_mode* contains the computed imaginary part of the e-ph self-energy (where phonon modes are numbered for increasing energy values).

The following is the format of *'prefix'.imsigma* (in this case, *si.imsigma*):

```
#  Electron (Imaginary) Self-Energy in the Migdal Approx.  #
#      ( only for bands within [band_min, band_max] )      #
#-----------------------------------------------------------#
# NO.k:    450   NO.bands:   2   NO.T:   1   NO.modes:   1
#
# Temperature(T)= 25.85203 meV;  Chem.Pot.(mu)= 6.55048 eV
#===========================================================
# it     ik   ibnd    E(ibnd)(eV)     Im(Sigma)(meV)
#-----------------------------------------------------------
  1       1     1      6.955370   1.2413716479777598E+01
  ......
  ......
#-----------------------------------------------------------
```

The variable *it* is a dummy variable for enumerating the temperature values, while, *ik* is the number of \(\mathbf{k}\) points in the fklist, *ibnd* the band number (in this case, band indices are 5 and 6). *Im(Sigma)* is the imaginary part of the e-ph self-energy (in meV units) for each state of interest.

Similarly, the format for *si.imsigma_mode* is

```
#  Electron (Imaginary) Self-Energy in the Migdal Approx.  #
#      ( only for bands within [band_min, band_max] )      #
#-----------------------------------------------------------#
# NO.k:    450   NO.bands:   2   NO.T:   1   NO.modes:   6
#
# Temperature(T)= 25.85203 meV;  Chem.Pot.(mu)= 6.55048 eV
#===========================================================
# it     ik   ibnd    E(ibnd)(eV)  imode    Im(Sigma)(meV)
#-----------------------------------------------------------
  1      1     1      6.955370     1   1.415350400936959E+00
  ......
  ......
#-----------------------------------------------------------
```

Here we have an extra column with the phonon mode index (*imode*).

> **Note:** One should always check the convergence of the e-ph self-energy with respect to the number of $\mathbf{q}$ points and the smearing parameter (delta_smear). Check this paper for more detail.

Using the results in the *'prefix'.imsigma* file, one can easily obtain, with a small script, the scattering rates for each state, which are equal to $\frac{2}{\hbar} \operatorname{Im}\Sigma$ (it's convenient to use $\hbar = 0.65821195\,\mathrm{eV}\,\mathrm{fs}$ to this end). Using additional tools provided in `perturbo.x`, we can also compute the mean free path for each electronic state, as well as a range of phonon-limited transport properties.

One way of obtaining the relaxation times (and their inverse, the scattering rates) is to run the Python script `relaxation_time.py` we provide to post-process the imsigma output (the desciption of the script is here). Another way is to obtain the relaxation times is to run a calculation of the mean free paths (see below), which conveniently outputs both the relaxation times and the mean free path for the desired electronic states.

Also note that an example calculation of the e-ph self-energy for holes, is provided in the example folder *"example02-silicon-perturbo/perturbo/pert-imsigma-hole"*, link, where we use different band indices (`band_min=2` and `band_max=4`), and the files, *fklist* and *ftemper*, are also different and obtained in a different perturbo `'setup'` calculation.

Electron mean free path
calc_mode = 'meanfp'

Directory: *example02-silicon-perturbo/perturbo/pert-meanfp-electron/*
link

Computes: The e-ph mean free paths for electronic states in a user-defined \(\mathbf{k}\) point list and range of bands.

Note: The mean free path calculation relies on the results of the calculation mode `'imsigma'` values obtained. Therefore, the user should first run the calculation mode `'imsigma'`, and then compute the mean free paths

Requires the same files as `calc_mode='imsigma'` but needs an additional file, *'prefix'.imsigma*, obtained as an output in the `'imsigma'` calculation.

Here is the input file (*pert.in*). It should be the same input as the one for the `'imsigma'` calculation mode, except for the line specifying `calc_mode='meanfp'`:

```
&perturbo
 prefix      = 'si'
 calc_mode   = 'meanfp'

 fklist   = 'si_tet.kpt'
 ftemper  = 'si.temper'

 band_min = 5
 band_max = 6

 phfreq_cutoff = 1 ! meV
 delta_smear = 10 ! meV

 sampling = 'uniform'
 nsamples = 1000000
/
```

Before running `perturbo.x`, make sure you have the following files in the current directory (*"pert-meanfp-electron"*): *'prefix'_epwan.h5*, *'prefix'.imsigma* the *fklist* file (*si_tet.kpt* in this example), and the *ftemper* file (e.g., *si.temper* in this example). As explained above, one can reuse the input file of the calculation mode `'imsigma'` by replacing the calculation mode with `calc_mode='meanfp'`.

```
$ mpirun -n 1 perturbo.x -npools 1 -i pert.in > pert.out
```

This calculation usually takes only takes a few seconds. We obtain two output files:

- *'prefix'.mfp* contains the relaxation time and mean free path of each electronic state. Note that the MFP is the product of the state relaxation time and the absolute value of the band velocity.
- *'prefix'.vel* contains the band velocity of each state

The format of *'prefix'.mfp* is as follows:

```
#============================================================#
#    Electron Mean Free Path (tau_nk * |v_nk|, in nm)      #
#============================================================#
#          NO.k:    2637   NO.bands:    2   NO.T:    1
#########
# Temperature(T)= 25.85203 meV;  Chem.Pot.(mu)= 6.55048 eV
#------------------------------------------------------------
# it ik ibnd E(ibnd)(eV) Relaxation time(in fs)  MFP (in nm)
#------------------------------------------------------------
  1   1  1   6.955370   2.6511488462206518E+01   9.592957354201
9302E+00
  ......
  ......
```

The variable *it* is the dummy variable for temperature; in this case, we only used one temperature (300 K). *ik* is the dummy variable for the given crystal momentum in the file fklist. *ibnd* is the dummy variable for bands; in this case, ibnd=1 corresponds to band index 5 and ibnd=2 is the band index 6. The 4th, 5th, and 6th columns are energy (eV), relaxation time (fs), and mean free path (nm) of each state, respectively.

The format of *'prefix'.vel* is shown below:

```
#######################################################
#                    Band velocity                    #
#######################################################
#  ik  ibnd   E(ibnd)(eV)     k.coord. (cart. ala
t)                vel-dir                  |vel| (m/s)
   1     1      6.955370  -0.01250  0.58750 -0.01250     -0.249
26 -0.93581 -0.24926   3.6184152269976016E+05
......
......
```

The 1st to 3rd columns are the same as in *'prefix'.mfp*. The 4th to 6th columns are the \(\mathbf{k}\) point coordinates in the crystal units. The 7th to 9th columns are the components of the unit vector specifying the direction of the velocity of each electronic states. The last column is the magnitude of the velocity (m/s) of each state.

For an example calculation of mean free paths for holes, please see the folder *"example02-silicon-perturbo/perturbo/pert-meanfp-hole"*, link .

Phonon-limited Carrier Transport

In this section of the tutorial, we will compute the electrical conductivity, carrier mobility tensors as well as the Seebeck coefficient and thermal conductivity. PERTURBO can compute these quantities using the relaxation time approximantion (RTA) of the Boltzmann transport equation (BTE): `calc_mode = 'trans-rta'`.

Another, more accurate, but also more expensive method is the iterative approach (ITA) to fully solve the linearized BTE: `calc_mode = 'trans-ita'`. The RTA and ITA calculations can be carried out in the presence of magnetic field: `calc_mode = 'trans-mag-rta'` and `calc_mode = 'trans-mag-ita'`.

The methodology of the **non-magnetic** transport calculations is described in this paper: *Comput. Phys. Commun.* **264**, 107970, (2021)

The methodology of the transport calculations in a **finite magnetic field** are presented here: *Phys. Rev. B***103**, L161103, (2021)

Relaxation time approximation (RTA)
calc_mode = 'trans-rta'

Directory: *example02-silicon-perturbo/perturbo/pert-trans-RTA-electron/* link

Computes: The phonon-limited conductivity, mobility, Seebeck coefficient and thermal conductivity using the RTA of the BTE at zero magnetic field.

Note: The user needs to run the calculation modes `'setup'` and then `'imsigma'` since this calculation mode relies on their outputs. If the *'prefix'.imsigma* file is absent, the calculation will proceed by computing scattering rates on the fly, which is more computationally expensive.

Requires the same variables as those specified in the calculation mode `'setup'`, except for the following two variables:

- calc_mode: set to `'trans-rta'` or `'trans'`

Here is the input file (*pert.in*):

```
&perturbo
 prefix      = 'si'
 calc_mode   = 'trans-rta'

 boltz_kdim(1) = 80
 boltz_kdim(2) = 80
 boltz_kdim(3) = 80

 boltz_emin = 6.4
 boltz_emax = 6.9
 band_min = 5
 band_max = 6

 ftemper  = 'si.temper'

 boltz_nstep = 0  ! optional
/
```

Before running `perturbo.x`, remember to put the following files in the run directory:

- *'prefix'_epwan.h5*: here *si_epwan.h5*
- *ftemper*: here *si.temper* obtained in the `'setup'` calculation
- *'prefix'_tet.h5*: here *si_tet.h5* obtained in the `'setup'` calculation
- *'prefix'.imsigma*: here *si.imsigma* obtained in the `'imsigma'` calculation

Run `perturbo.x`:

```
$ mpirun -n 1 perturbo.x -npools 1 -i pert.in > pert.out
```

This calculation usually takes a few minutes. We obtain five output files:

- *'prefix'.cond* contains the conductivity and mobility tensors for each configuration in the temper file
- *'prefix'.tdf* contains transport distribution function (TDF) as a function of carrier energy and temperature
- *'prefix'_tdf.h5* includes all the information of the TDF and occupation

changes for each configuration in HDF5 format

- *'prefix'.trans_coef* contains the conductivity, mobility, Seebeck coefficient and thermal conductivity tensors
- *pert_output.yml* is the YAML output file containing information about the input and output parameters as well as the conductivity and mobility

In our example, the output file is *si.cond*, which is shown here:

```
#=========================================================#
#                  Conductivity (1/Ohm/m)                 #
#---------------------------------------------------------#

#  T (K)   E_f(eV)   n_c (cm^-3)      sigma_xx       sigma_x
y       sigma_yy       sigma_xz       sigma_yz       sigma_zz
  300.00   6.55048   0.99458E+18    0.235555E+05  -0.744950E-0
6   0.235555E+05  -0.126413E-06  -0.247918E-04   0.235555E+05


#=========================================================#
#                    Mobility (cm^2/V/s)                  #
#--------------------(for semiconductor)------------------#

#  T (K)   E_f(eV)   n_c (cm^-3)       mu_xx          mu_x
y          mu_yy          mu_xz          mu_yz          mu_zz
  300.00   6.55048   0.99458E+18    0.147822E+04  -0.467493E-0
7   0.147822E+04  -0.793302E-08  -0.155581E-05   0.147822E+04
```

The calculated electron mobility at 300 K is ~ 1478 $cm^2V^{-1}s^{-1}$, in reasonably good agreement with the experimental value of roughly 1400 $cm^2V^{-1}s^{-1}$.

The second output file is *si.tdf*, whose format is shown below:

```
 #  E(eV)    (-df/dE) (a.u.)    TDF(E)_(xx xy yy xz yz zz)
(a.u.)   #

 # Temperature:  300.0000  Chemical Potential:   6.550482

    6.632612    2.0230556858340698E+01    0.000000E+00   0.0000
00E+00   0.000000E+00   0.000000E+00   0.000000E+00   0.000000
E+00
   ......
   ......
```

Column 1 is the carrier energy (eV), column 2 is the energy derivative of Fermi-Dirac distribution at the energy given by column 1, and columns 3-8 are the TDF values for each energy (same as conductivity, TDF has six components, usually the longitudinal component is plotted), respectively. The data for each temperature and chemical potential combination is given in a separate block of the file. In this case, we look at one temperature and one concentration, so there is only one block in the file.

Please refer to the calc_mode= `'trans-pp'` to see the description of the *'prefix'.trans_coef* output file.

In more rigorous calculations, the user will need to converge the conductivity and mobility with respect to the number of $\mathbf{k}$ and $\mathbf{q}$ points, namely the variables boltz_kdim and boltz_qdim.

An example for hole carriers is also provided, in the folder *"example02-silicon-perturbo/perturbo/pert-trans-RTA-hole"* .

Full solution: Iterative approach (ITA)

calc_mode = 'trans-ita'

Directory: *example02-silicon-perturbo/perturbo/pert-trans-ITA-electron/* link

Computes: The phonon-limited conductivity, mobility, Seebeck coefficient and thermal conductivity tensors iteratively (ITA) at zero magnetic field.

Note: The user needs to run the calculation modes `'setup'` since this

calculation mode relies on their outputs. The *'prefix'.imsigma* file is optional, use it as a starting point for the iterative process if present.

Requires the same input file variables as the calculation mode `'setup'`, except for the following 6 variables:

- calc_mode: is set to `'trans-ita'`
- boltz_nstep: contains the maximum number of iterations in the iterative scheme for solving Boltzmann equation, where a typical value is 10
- phfreq_cutoff: contains phonon threshold (meV). Phonons with energy smaller than the cutoff will be ignored.
- delta_smear: contains broadening (meV) for a Gaussian function to present the Dirac delta function
- tmp_dir: contains output directory containing the e-ph matrix elements used in the calculations
- load_scatter_eph: if `.true.`, it will read the e-ph matrix elements from *tmp_dir*. The default is `.false.`

Here is the input file (*pert.in*):

```
&perturbo
 prefix      = 'si'
 calc_mode   = 'trans-ita'

 boltz_kdim(1) = 80
 boltz_kdim(2) = 80
 boltz_kdim(3) = 80

 boltz_emin = 6.4
 boltz_emax = 6.9
 band_min = 5
 band_max = 6

 ftemper  = 'si.temper'

 tmp_dir = './tmp'
 !load_scatter_eph = .true.

 boltz_nstep = 10 !max number of iterations
 phfreq_cutoff = 1  !meV
 delta_smear = 10  !meV
/
```

Before running the ITA calculation, make sure that the following files are in the run directory :

- *'prefix'_epwan.h5*: here *si_epwan.h5*
- *ftemper*: here *si.temper*
- *'prefix'_tet.h5*: here *si_tet.h5*

```
export OMP_NUM_THREADS=4
$ mpirun -n 8 perturbo.x -npools 8 -i pert.in > pert.out
```

This task is time-consuming using one thread and one MPI process on a single core. To speed up the calculations, we run it on multiple cores using hybrid MPI plus OpenMP parallelization. After the calculation has completed, we obtain 5 output files, *'prefix'.cond*, *'prefix'.tdf*, *'prefix'_tdf.h5*, *'prefix'.trans_coef*, and *pert_output.yml*, similar to the RTA calculation.

> **Note:** For ITA calculations, each MPI process could consume a significnat amount of RAM (memory). If RAM of computing nodes is limited, one can set

`OMP_NUM_THREADS` to the total number of cores of the computing node, and set the MPI process per node to 1.

An example calculation for holes is also provided in the folder *"example02-silicon-perturbo/perturbo/pert-trans-ITA-hole"*, link .

Magnetic RTA

calc_mode = 'trans-mag-rta'

Directory: *example02-silicon-perturbo/perturbo/pert-trans-mag-RTA-electron/* link

Computes: The phonon-limited conductivity and carrier mobility using RTA in a finite magnetic field.

Note: The user needs to run the calculation modes `'setup'` since this calculation mode relies on their outputs. The *'prefix'.imsigma* file is optional, but speeds up the calculation significantly since otherwise the matrix elements and scattering rates are computed on the fly.

Requires the same input file variables as the calculation mode `'trans-rta'`, except for the following variables:

- calc_mode: is set to `'trans-mag-rta'`
- boltz_nstep: Note that in the magnetic Boltzmann equation, RTA requires a finite number of iterations.

Here is the input file (*pert.in*):

```
&perturbo
 prefix      = 'si'
 calc_mode   = 'trans-mag-rta'

 boltz_kdim(1) = 80
 boltz_kdim(2) = 80
 boltz_kdim(3) = 80

 boltz_emin = 6.4
 boltz_emax = 6.9
 band_min = 5
 band_max = 6

 ftemper  = 'si.temper'

 boltz_nstep = 10 !max number of iterations
/
```

Before running the calculation, make sure that the following files are in the run directory :

- *'prefix'_epwan.h5*: here *si_epwan.h5*
- *ftemper*: here *si.temper*
- *'prefix'_tet.h5*: here *si_tet.h5*
- *'prefix'.imsigma*: here *si.imsigma* (optional but recommended for faster calculations)

The *'prefix'.temper* file needs to be modified to include magnetic fields. Add the \(x\), \(y\), and \(z\) values of magnetic fields (in Tesla) in the last three columns of the temper file.

```
1 F                                  ! Bx   By   Bz
300.00   6.5504824219   0.9945847E+18  0.00 0.00 0.01
```

Run PERTURBO:

```
export OMP_NUM_THREADS=4
$ mpirun -n 8 perturbo.x -npools 8 -i pert.in > pert.out
```

We run the calculation on multiple cores using hybrid MPI plus OpenMP parallelization. After the calculation has completed, we obtain 5 output files, *'prefix'.cond*, *'prefix'.tdf*, and *'prefix'_tdf.h5*, *'prefix'.trans_coef*, and *pert_output.yml* .

Because all elements of the conductivity tensor are independent in a non-zero magnetic field, the *'prefix'.cond* file prints out all 9 elements of the conductivity and mobility tensors.

```
#==========================================================#
          #                  Conductivity (1/Ohm/
m)                  #

#----------------------------------------------------------#

#  T (K)   E_f(eV)   n_c (cm^-3)      sigma_xx       sigma_x
y       sigma_yy       sigma_xz       sigma_yz       sigma_z
z       sigma_yx       sigma_zx       sigma_zy
  300.00   6.55048   0.99458E+18    0.235554E+05  -0.353835E+0
2   0.235554E+05   0.169392E-04  -0.324052E-04   0.235555E+0
5   0.353972E+02   0.235280E-02  -0.101837E-02
          #--------------------iterative proces
s---------------------#
  #iter.       sigma_xx       sigma_xy       sigma_yy       sig
ma_xz       sigma_yz       sigma_zz       sigma_yx       sigm
a_zx       sigma_zy
  #   1      0.235555E+05  -0.744950E-06   0.235555E+05  -0.126
413E-06  -0.247918E-04   0.235555E+05  -0.744950E-06  -0.126413
E-06  -0.247918E-04
  #   2      0.235554E+05  -0.353835E+02   0.235554E+05   0.169
392E-04  -0.324052E-04   0.235555E+05   0.353972E+02   0.235280
E-02  -0.101837E-02

#----------------------------------------------------------#




#==========================================================#
          #                     Mobility (cm^2/V/
s)                  #
          #--------------------(for semiconducto
r)-------------------#

#  T (K)   E_f(eV)   n_c (cm^-3)       mu_xx          mu_x
y          mu_yy          mu_xz          mu_yz          mu_z
z          mu_yx          mu_zx          mu_zy
  300.00   6.55048   0.99458E+18    0.147822E+04  -0.222049E+0
1   0.147822E+04   0.106302E-05  -0.203359E-05   0.147822E+0
4   0.222135E+01   0.147650E-03  -0.639081E-04
```

In a magnetic calculation, the Seebeck and thermal conductivity tensors are not computed. Hence, the *'prefix'.trans_coef* file only contains the conductivity and mobility tensors.

```
#==========================================================#
          #                  Conductivity (1/Ohm/
m)                  #

#----------------------------------------------------------#

#  T (K)   E_f(eV)   n_c (cm^-3)      sigma_xx       sigma_x
y       sigma_yy       sigma_xz       sigma_yz       sigma_z
z       sigma_yx       sigma_zx       sigma_zy
  300.00   6.55048   0.99458E+18    0.235554E+05  -0.353835E+0
2   0.235554E+05   0.169392E-04  -0.324052E-04   0.235555E+0
5   0.353972E+02   0.235280E-02  -0.101837E-02




#==========================================================#
          #                    Mobility (cm^2/V/
s)                  #
          #--------------------(for semiconducto
r)-------------------#

#  T (K)   E_f(eV)   n_c (cm^-3)       mu_xx          mu_x
y          mu_yy          mu_xz          mu_yz          mu_z
z          mu_yx          mu_zx          mu_zy
  300.00   6.55048   0.99458E+18    0.147822E+04  -0.222049E+0
1   0.147822E+04   0.106302E-05  -0.203359E-05   0.147822E+0
4   0.222135E+01   0.147650E-03  -0.639081E-04
```

Magnetic ITA

calc_mode = 'trans-mag-ita'

🗀 **Directory:** *example02-silicon-perturbo/perturbo/pert-trans-mag-ITA-electron/* link

Computes: The phonon-limited conductivity and carrier mobility using full Boltzmann equation in a finite magnetic field.

Note: The user needs to run the calculation modes `'setup'` since this calculation mode relies on their outputs. The *'prefix'.imsigma* file is optional, use it as a starting point for the iterative process if present.

Requires the same input file variables as the calculation mode `'trans-ita'`, except for the following variable:

- calc_mode: set to `'trans-mag-ita'`

Here is the input file (*pert.in*):

```
&perturbo
 prefix      = 'si'
 calc_mode   = 'trans-mag-ita'

 boltz_kdim(1) = 80
 boltz_kdim(2) = 80
 boltz_kdim(3) = 80

 boltz_emin = 6.4
 boltz_emax = 6.9
 band_min = 5
 band_max = 6

 ftemper  = 'si.temper'

 tmp_dir = './tmp'
 !load_scatter_eph = .true.

 boltz_nstep = 10 !max number of iterations
 phfreq_cutoff = 1  !meV
 delta_smear = 10  !meV
/
```

Before running the calculation, make sure that the following files are in the run directory :

- *'prefix'_epwan.h5*: here *si_epwan.h5*
- *ftemper*: here *si.temper*

- *'prefix'_tet.h5*: here *si_tet.h5*
- *'prefix'.imsigma*: here *si.imsigma* (optional)

Like in the calculation mode `'trans-mag-rta'`, the *'prefix'.temper* file needs to be modified to include magnetic fields. Add the \(x\), \(y\), and \(z\) values of magnetic fields (in Tesla) in the last three columns of the temper file.

```
1 F                                     !Bx   By   Bz
300.00   6.5504824219   0.9945847E+18  0.00 0.00 0.01
```

Run PERTURBO:

```
export OMP_NUM_THREADS=4
$ mpirun -n 8 perturbo.x -npools 8 -i pert.in > pert.out
```

We run the calculation on multiple cores using hybrid MPI plus OpenMP parallelization. After the calculation has completed, we obtain 5 output files, *'prefix'.cond*, *'prefix'.tdf*, and *'prefix'_tdf.h5*, *'prefix'.trans_coef*, and *pert_output.yml* . The format of the output files is the same as in the calculation mode `'trans-mag-rta'`.

Postprocessing of transport

calc_mode = 'trans-pp'

Directory: *example02-silicon-perturbo/perturbo/pert-trans-pp-electron/*

link

Computes: Seebeck coefficient and thermal conductivity for the **non-magnetic case**. Note that phonon drag effects are not included in this calculation.

Note: The Seebeck and thermal conductivity are also computed in the non-magnetic `'trans'` calculations. However, since this step is very quick, we provide a dedicated calculation mode for the case of a manual modifications of the *'prefix'_tdf.h5* or other scenarios when only this step is required.

Uses the similar input file to the `'trans'` calculation modes, but requires the additional file *'prefix'_tdf.h5* obtained in the `'trans'` calculation.

Change the calculation mode in the input file to `'trans-pp'`. Before running `perturbo.x`, make sure that four files exist in the current directory:

- *'prefix'_epwan.h5*: here *si_epwan.h5*
- *ftemper*: here *si.temper*
- *'prefix'_tet.h5*: here *si_tet.h5*
- *'prefix'_tdf.h5*: here *si_tdf.h5*

Run `perturbo.x`:

```
$ mpirun -n 1 perturbo.x -npools 1 -i pert.in > pert.out
```

It takes a few seconds. We obtain a file, *'prefix'.trans_coef*, in this case, si.trans_coef, which has the following format:

```
#============================================================#
         #                  Conductivity (1/Ohm/
m)                  #

#------------------------------------------------------------#

#  T (K)   E_f(eV)   n_c (cm^-3)      sigma_xx       sigma_x
y       sigma_yy       sigma_xz       sigma_yz       sigma_zz
  300.00   6.55048   0.99458E+18    0.251810E+05  -0.106635E+0
0   0.251823E+05  -0.172325E+00   0.142428E+00   0.251812E+05


#============================================================#
         #                    Mobility (cm^2/V/
s)                   #
         #--------------------(for semiconducto
r)-------------------#

#  T (K)   E_f(eV)   n_c (cm^-3)       mu_xx          mu_x
y          mu_yy          mu_xz          mu_yz          mu_zz
  300.00   6.55048   0.99458E+18    0.158023E+04  -0.669186E-0
2   0.158031E+04  -0.108143E-01   0.893806E-02   0.158025E+04


#============================================================#
          #                Seebeck coefficient (mV/
K)                #

#------------------------------------------------------------#

#  T (K)   E_f(eV)   n_c (cm^-3)        S_xx           S_x
y           S_yy           S_xz           S_yz           S_zz
  300.00   6.55048   0.99458E+18   -0.428128E+00  -0.125064E-0
6  -0.428127E+00   0.500669E-07  -0.531919E-07  -0.428128E+00


#============================================================#
          #                Thermal conductivity (W/m/
K)              #
          #------------------(Electronic contributio
n)---------------#

#  T (K)   E_f(eV)   n_c (cm^-3)      kappa_xx       kappa_x
y       kappa_yy       kappa_xz       kappa_yz       kappa_zz
```

```
  300.00   6.55048   0.99458E+18    0.686130E-01  -0.143531E-0
6   0.686131E-01  -0.142782E-06  -0.114048E-07   0.686144E-01
```

The two blocks for the conductivity and mobility are the same as those in the `'trans'` calculation mode, but the output file of `'trans-pp'` has an additional block with the Seebeck coefficient results.

An example calculation for holes is also provided in the folder *"example02-silicon-perturbo/perturbo/pert-trans-pp-hole"* .

Ultrafast Dynamics

In this section, we describe the ultrafast carrier dynamics. In contrast to other PERTURBO calculation modes, here we solve the real-time Boltzman transport equation (rt-BTE):

$$\frac{\partial f_{n\mathbf{k}}(t)}{\partial t} = \frac{e\mathbf{E}}{\hbar}\cdot \nabla _\mathbf{k}f_{n\mathbf{k}}(t)+\mathcal{I}^{e-\mathrm{ph}}[f_{n\mathbf{k}}(t)],$$

where $f_{n\mathbf{k}}(t)$ is the time-dependent electronic occupation of a Bloch state with band index n and crystal momentum $\mathbf{k}$, and e is the electronic charge. The first term on the right-hand side, called the advection term, is proportional to the external electric field $\mathbf{E}$ and is responsible for electron drift. The second term is the e-ph collision integral $\mathcal{I}^{e-\mathrm{ph}}$ accounting for phonon absorption and emission processes.

First, we will consider the zero field ultrafast dynamics ($\mathbf{E}=0$), then the high-field dynamics (full rt-BTE) will be considered.

The methodology of the **zero-field** ultrafast dynamics implemented in PERTURBO can be found in these papers: *Nano Lett.* **17**, 5012-5019, (2017) , *Comput. Phys. Commun.* **264**, 107970, (2021)

The methodology of the **high-field** ultrafast dynamics implemented in PERTURBO can be found in this paper: *Phys. Rev. B***104**, L100303, (2021)

Zero-field ultrafast dynamics
calc_mode = 'dynamics-run'

Directory: *example02-silicon-perturbo/perturbo/pert-dynamics-run/* link

Computes: Ultrafast hot carrier dynamics via the time-dependent Boltzmann transport equation: set an initial carrier distribution and calculate its evolution in time at zero external fields.

For the ultrafast dynamics, one needs first to perform the `'setup'` calculation. So, we assume that the user has already performed this calculation. From the `'setup'` calculation, we retain the following files necessary for the dynamics calculations: *'prefix'.temper* and *'prefix'_tet.h5*.

> ℹ **Note:** For the ultrafast dynamics we will need a larger energy window. So, the `'setup'` calculation should be performed with `boltz_emin = 6.4` and `boltz_emax = 7.4`.

In a typical zero-field dynamics, at $t=0$, one excites the electronic system with a Gaussian pulse and then follows the electron relaxation due to the electron-phonon scattering. The excited carrier concentration is preserved during the real-time simulation and the integral of the excitation pulse corresponds to the excited carrier density.

> ℹ **Note:** For the ultrafast dynamics, the carrier density is determined by the initial condition and not by the *'prefix'.temper* file, in contrast to other calculation modes.

For the `'dynamics-run'` calculation, specify the following variables in the input file (*pert.in*):

- preifx: the same prefix as in *'prefix'_epwan.h5*
- calc_mode: set to `'dynamics-run'`
- boltz_kdim: $\mathbf{k}$ grid for electrons, here we use a 80x80x80 grid
- boltz_qdim: $\mathbf{q}$ grid for phonons, specify if it is different from the $\mathbf{k}$ grid, here we use the same $\mathbf{q}$ grid as $\mathbf{k}$ grid
- boltz_emin, boltz_emax: energy window (in eV units), use the same as in the `'setup'` calculation (here, 6.4 and 7.4 eV)
- band_min, band_max: band range
- ftemper: the filename of a file containing temperature, chemical potential, and carrier concentration values (see the format)
- time_step: simulation time step Δt, set to its typical value, 1 fs
- boltz_nstep: total number of time steps, set to 50; here we perform a relatively short simulation of 50 fs
- output_nstep: an optional variable to shorten the output; the output time step Δt_{out} is determined in the following way: $\Delta t_{out} = \texttt{output_nstep}\times \Delta t$
- solver: BTE solver type, set to `'euler'`, here we use the Euler first order solver of BTE
- boltz_init_dist: set to `'gaussian'`, we select the Gaussian initial

distribution. To restart the simulation, specify `boltz_init_dist='restart'`, then the distribution of the last step from the previous simulation will be used.

- boltz_init_e0: in this example, the Gaussian distribution is centered around 7.0 eV
- boltz_init_smear: we select a 40 meV smearing
- phfreq_cutoff: we select a 1 meV phonon energy cutoff
- delta_smear: the broadening to model the Dirac delta function is chosen to 8 meV

Here is the input file (*pert.in*):

```
&perturbo
 prefix      = 'si'
 calc_mode   = 'dynamics-run'

 boltz_kdim(1) = 80
 boltz_kdim(2) = 80
 boltz_kdim(3) = 80

 boltz_emin = 6.4
 boltz_emax = 7.4
 band_min = 5
 band_max = 6

 ftemper  = 'si.temper'

 time_step   = 1 !fs
 boltz_nstep = 50
 output_nstep = 2
 solver = 'euler'

 boltz_init_dist = 'gaussian'
 boltz_init_e0 = 7.0 ! eV
 boltz_init_smear = 40 !meV

 tmp_dir = "./tmp"
 phfreq_cutoff = 1 ! meV
 delta_smear = 8 ! meV
/
```

In this example, we calculate the evolution of the electron distribution. In order to perform the hole dynamics, set the parameter hole to `true`.

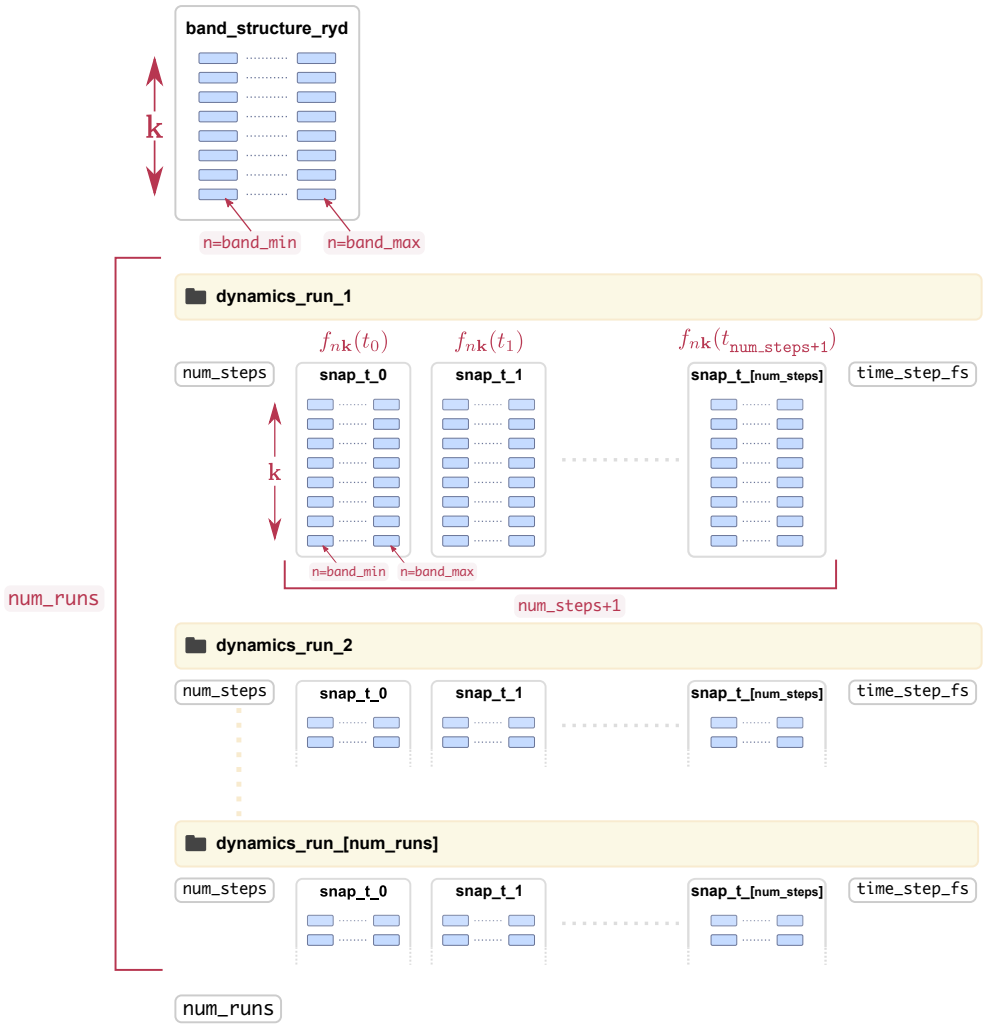
Run `perturbo.x` (remember to link or copy *'prefix'_epwan.h5* in the current directory):

```
$ export OMP_NUM_THREADS=4
$ mpirun -n 8 <perturbo_bin>/perturbo.x -npools 8 -i pert.in >
pert.out
```

We obtain the *'prefix'_cdyna.h5* HDF5 output file (this file can be also found in the *"References"* directory). This file contains all the necessary output information about the performed simulation. This file is organized as follows:

- `band_structure_ryd`: electronic bandstructure in Ry; each column corresponds to the band index \(n\) \((~\texttt{band_min}\leq n \leq \texttt{band_max})\)
- `dynamics_run_[i]`: an HDF5 group that contains information about the ith simulation.
  If the simulation was restarted (`boltz_init_dist='restart'`) one or more times, one will have several dynamics_run_[i] groups, otherwise, only dynamics_run_1 will be present. A group dynamics_run_[i] is structured as follows:
  - `num_steps`: the number of *output* time steps (taking into account output_nstep), can be different for different dynamics_run_[i]
  - `snap_t_0`: \(f_{n\mathbf{k}}(t_0)\)
    \(\vdots\)
  - `snap_t_[j]`: the distribution function \(f_{n\mathbf{k}}\) for time \(t_j\): \(t_j = t_0 + j\Delta t_{out}\), where \(\Delta t_{out}\) is the output time step. Each column of the array corresponds to the band index \(n\)
    \(\vdots\)
  - `snap_t_[num_steps]`: \(f_{n\mathbf{k}}(t_{\texttt{num_steps}})\)
  - `time_step_fs`: the output time step \(\Delta t_{out}\) (can be different for different dynamics_run_[i])
- `num_runs`: total number of performed simulations (corresponds to the number of dynamics_run_[i] groups).

The *'prefix'_cdyna.h5* file structure can be schematically represented as follows:



The HDF5 files can be easily processed by Python package h5py . As an example, we present here a simple Python script that visualizes the distribution function for the time \(t_5\) of the simulation and for the first band (in the selected band range):

```
#!/usr/bin/env python3
import h5py
import matplotlib.pyplot as plt

prefix='si'
snap_number=5
band_index=0

# load the HDF5 file
h5file = h5py.File(prefix+'_cdyna.h5', 'r')

# get the data
ryd2ev = h5file['band_structure_ryd'].attrs['ryd2ev']
energy_ev = h5file['band_structure_ryd'][:,band_index] * ryd2ev
dist_func = h5file['dynamics_run_1']['snap_t_'+str(snap_numbe
r)][:,band_index]
h5file.close()

# plot the data
plt.plot(energy_ev,dist_func,marker='o',linestyle='')
plt.xlabel('Energy (eV)')
plt.ylabel('Distribution function')
plt.show()
```

In order to postprocess this file using `perturbo.x`, see the next section.

Dynamics post-processing
calc_mode = 'dynamics-pp'

Directory: *example02-silicon-perturbo/perturbo/pert-dynamics-pp/* link

Computes: Postprocessing of the ultrafast dynamics calculations: carrier population as a function of energy and time.

In this section we aim to calculate the Brillouin zone-averaged *energy*-dependent carrier population $\bar{f}(E,t)$. Having calculated the distribution function $f_{n\mathbf{k}}(t)$, one can find $\bar{f}(E,t)$ in the following way:

$$\bar{f}(E,t) = \sum_{n\mathbf{k}} f_{n\mathbf{k}}(t) \delta(\epsilon_{n\mathbf{k}}-E).$$

The integral of $\bar{f}(E,t)$ over the energy gives the number of carriers per unit cell as a function of time.

In order to calculate the $\bar{f}(E,t)$ quantity, one needs to have all the files required for the `calc_mode='dynamics-run'` calculation (previous section) and the HDF5 output file *'prefix'_cdyna.h5* from the dynamics-run calculation. To perform the postprocessing, use a similar to the previous section input file, but change the calculation mode to `calc_mode='dynamics-pp'`. Run `perturbo.x` (remember to link or copy *'prefix'_epwan.h5* in the current directory):

```
$ mpirun -n 1 <perturbo_bin>/perturbo.x -npools 1 -i pert.in >
pert.out
```

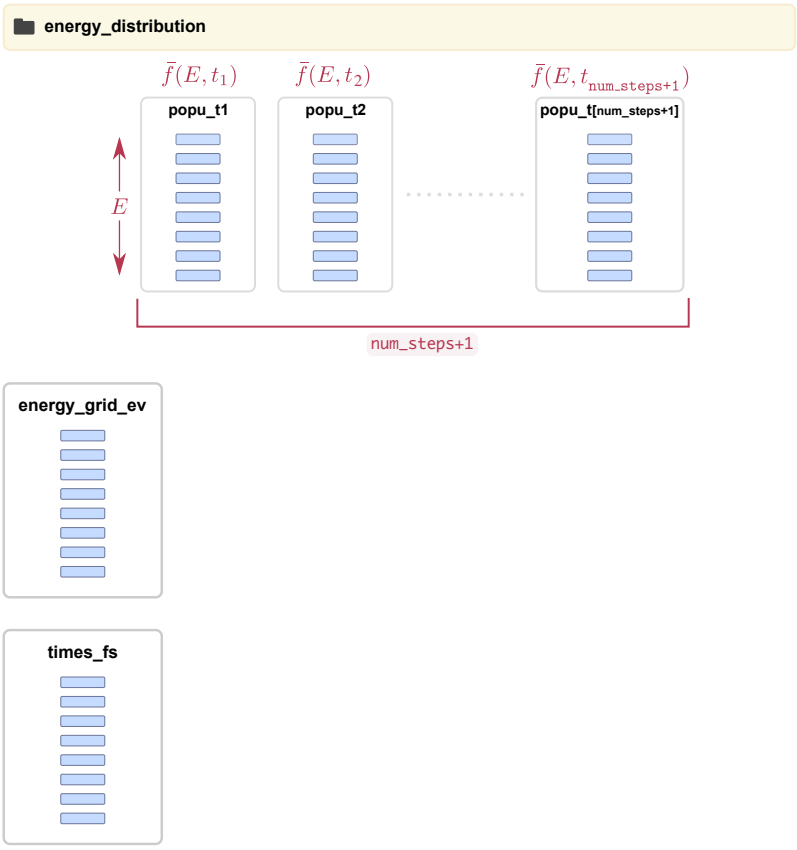
On the output, we obtain the following files:

- *si_popu.h5*: an HDF5 file that contains all the necessary information for $\bar{f}(E,t)$
- *si_cdyna.dat*: an ASCII file containing the number of carriers per unit cell as a function of time

The *si_popu.h5* HDF5 file is organized as follows:

- `energy_distribution`: a group that contains the populations for all the time instants of the dynamics-run simulation
    - `popu_t1`: $\bar{f}(E,t_1)$ $\vdots$
    - `popu_t[j]`: the carrier population $\bar{f}(E,t_j)$ at time t_j $\vdots$
    - `popu_t[num_steps+1]`: $\bar{f}(E,t_{\texttt{num_steps+1}})$
- `energy_grid_ev`: the grid of energies in eV; the number of energy grid points is given by $\frac{ \texttt{emax} - \texttt{emin} }{ \texttt{boltz_de} }+\texttt{3}$
- `times_fs`: the array of time instants in fs

The *si_popu.h5* HDF5 file can be schematically represented as follows:



Similarly to the previous section, we provide here a simplistic Python script showing an example how to manipulate this HDF5 file. For example, to plot the electron population for the time \(t_{25}\), run:

```python
#!/usr/bin/env python3
import h5py
import matplotlib.pyplot as plt

prefix='si'
snap_number=25

# load the HDF5 file
h5file = h5py.File(prefix+'_popu.h5', 'r')

# get the data
energy_ev = h5file['energy_grid_ev'][()]
population = h5file['energy_distribution']['popu_t'+str(snap_nu
mber)][()]
h5file.close()

# plot the data
plt.plot(energy_ev,population,marker='o',linestyle='')
plt.xlabel('Energy (eV)')
plt.ylabel('Electron population')
plt.show()
```

It is also convenient to postprocess and visualize the data in HDF5 file using other high level languages, such as Julia. For example, the following Julia script does the same thing as the above Python script:

```julia
using HDF5, Plots

prefix = "si"
fname = prefix * "_popu.h5"
snap_number = 25

# read the data
energy_ev = h5read(fname, "energy_grid_ev")
population = h5read(fname, "energy_distribution/popu_t"*strin
g(snap_number))

# plot
plot(energy_ev, population, xlabel="Energy (eV)", ylabel="Elect
ron population")
```

Click here to see a video tutorial on this topic.

Interface to TDEP for anharmonic phonons

For materials with strong lattice anharmonicity, it is critical to incorporate anharmonic phonons in electron-phonon calculations, take SrTiO3 for example (see this paper). PERTURBO provide interface to the TDEP package for finite temperature lattice dynamics.

Compilation

> **Note:** The TDEP version we tested and recommend is the version committed on Jun 14, 2018, (commit #ef3d150). Newer versions might also work, but their compatibility with PERTURBO have not been extensively tested.

To turn on TDEP interface in PERTURBO, one needs to **compile** PERTURBO with a flag `-D__TDEP` and link with the TDEP library, which can be done by adding the following lines to `make.sys` of PERTURBO.

```
#For TDEP interface
TDEP_root= ## top directory of TDEP ##
FFLAGS  += -D__TDEP
IFLAGS  += -I${TDEP_root}/inc/libolle
LIBOBJS += ${TDEP_root}/lib/libolle.a
```

Usage

> **Directory:** *example07-sto-tdep* link

To incorporate TDEP interatomic force constants in PERTURBO, one needs to prepare two key files from TDEP, namely *infile.ucposcar* and *infile.forceconstant*, and set `tdep = .true.` in *qe2pert.in* (the main input file for `qe2pert.x`).

Here is an example of the input file for `qe2pert.x` with the TDEP force constants:

```
&qe2pert
 prefix = 'sto'
 outdir = './tmp'
 phdir = './phonon/References/save'
 nk1 = 4
 nk2 = 4
 nk3 = 4
 num_wann = 3
 dft_band_min = 21
 dft_band_max = 23
 lwannier = .true.
 tdep = .true.  ! flag to turn on tdep
/
```

Once the TDEP force constants are employed in *'prefix'_ephwan.h5*, `perturbo.x` will use them automatically, there is no need to modify the input file for `perturbo.x`.

In the PERTURBO-TDEP interface, `qe2pert.x` reads data from *infile.ucposcar* and *infile.forceconstant*, maps the TDEP force contants to the Wigner Seitz Supercell (WS) generated by PERTURBO, and then writes them to *'prefix'_ephwan.h5*. If any pair of force constants in *infile.forceconstant* cannot be mapped to WS, `qe2pert.x` will issue a runtime error. To solve this issue, one needs to either increase the coarse \(\mathbf{q}\)-grid or decrease the `rc2` parameter (the cutoff for the 2nd force constants) in TDEP.

Technical details

Treatment of long-range dipole-dipole corrections in TDEP and PERTURBO

For more details of the dipole-dipole corrections for polar materials, we refer the users to this paper and its Supplemental Material.

In both TDEP and PERTURBO, only the short-ranged forceconstants are stored in the form of interatomic force constants (IFC) for polar materials, while the long-ranged dipole-dipole contributions to the dynamical matrices are computed on the fly, using the dielectric tensor, Born effective charges, and Ewald parameters.

There are several types of dipole-dipole corrections implemeneted in TDEP. PERTURBO uses the polar correction method proposed in this paper , which is implemented in TDEP as the correction type `3`.

Here is an example to run TDEP:

```
extract_forceconstants -rc2 5 --polar -pc 3
```

Here, `--polar` means the dipole-dipole correction is on for polar materials, and `-pc 3` indicates using the polar correction scheme introduced in our 2018 PRL paper .

The *infile.forceconstant* file from TDEP contains the short-ranged forceconstants extracted by fitting the displacement and subtracted forces and the quantities to compute the long-ranged (dipole-dipole) forceconstant, such as the polar correction type, dielectric tensor, the Born effective charges, and so on.

This is an example of the *infile.forceconstant* file:

```
           3 # This is a forceconstant for a polar material.
       6.255018086086       0.000000000000       0.00000000000
0 Dielectric tensor xx xy xz
       0.000000000000       6.255018086086       0.00000000000
0 Dielectric tensor yx yy yz
       0.000000000000       0.000000000000       6.25501808608
6 Dielectric tensor zx zy zz
  0.99826965332031237       # Coupling parameter in Ewald summa
tion
           4  # number of irreducible components in the Born ch
arges
```

The number `3` in the first line indicates the polar correction type. The fifth line shows the Ewald summation parameters.

`qe2pert.x` will check the polar correction type. If PC is `0`, then the polar correction is turned off. If the correction type is `3`, polar correction is turned on, otherwise, `qe2pert.x` will issue a runtime error. If correction type is `3`, dielectric tensor, Born effective charnges, and Ewald parameter will be written to the *'prefix'_ephwan.h5* file.

For more details, including running TDEP with VASP and Quantum Espresso, the issue with Born effective charge, etc., please refer to this document.

Other Tutorials

Summary: In addition to the silicon example discussed above, we provide several tutorial examples to explore the various capabilities of Perturbo. Before starting this tutorial, please read the sections on qe2pert.x and perturbo.x of this manual.

For each example in the tutorial, we use three directories to organize the results of the calculations:

- *pw-ph-wan*: contains files for the scf, nscf, phonon, and Wannier90 calculations when running Quantum Espresso (QE)
- *qe2pert*: contains files for running `qe2pert.x` to generate an essentail file '*prefix'_epwan.h5* for perturbo calculations
- *perturbo*: contains files for running `perturbo.x`

As a reminder, here are the steps needed to compute *prefix_epwan.h5*:

- Step 1: scf calculation
- Step 2: phonon calculation
    - collect all the data into a directory called *"save"*
- Step 3: nscf calculation
- Step 4: Wannierization with Wannier90
- Step 5: run `qe2pert.x`
    - soft link *'prefix'_centres.xyz*, *'prefix'_u.mat* (and, when present, *'prefix'_u_dis.mat*) in the directory *"pw-ph-wann/wann"*
    - create a directory called *"tmp"*, and inside it soft link the QE nscf output directory *'prefix'.save* in the *"pw-ph-wann/nscf/tmp"*

Note: For each `perturbo.x` calculation, it is essential to always link or copy *'prefix'_epwan.h5*.

Silicon: Spin-Orbit Coupling

Directory: *example03-silicon-soc/* link

Run qe2pert.x and perturbo.x on silicon with spin-orbit coupling

The input files can be found in the directory *"pw-ph-wann"*. Remember to run scf, nscf and Wannier90 calculations that include spinor-related variables. Once the DFT and DFPT calculations are completed, we run `qe2pert.x` to generate *'prefix'_epwan.h5*. In the input file for `qe2pert.x` (*"qe2pert/pert.in"*) the user does not need to specify any spinor-related variables since `qe2pert.x` is able to detect that spin-orbit coupling (SOC) was used in DFT. Here is the input file (*pert.in*):

```
&qe2pert
 prefix='si'
 outdir='./tmp'
 phdir='../pw-ph-wann/phonon/References/save'
 nk1=8, nk2=8, nk3=8
 dft_band_min = 1
 dft_band_max = 32
 num_wann = 16
 lwannier=.true.
/
```

The input file is similar to the one for silicon without SOC (*"example02-silicon-qe2pert"*, link). We only need to double the number of Wannier functions (num_wann variable) and DFT bands (dft_band_min and dft_band_max) in the input file.

The input files for `perturbo.x` are also similar to the silicon calculations without SOC, except for the band range given by dft_band_min and dft_band_max. Each calculation is the same as in the silicon example without SOC.

GaAs: Polar Material

Directory: *example04-gaas-polar/* link

Example calculation in a polar material with long-range e-ph interactions.

Run the calculations in the directory *"pw-ph-wann"* to obtain the data needed to run `qe2pert.x`. Run `qe2pert.x` to get *'prefix'_epwan.h5*, which is required for all calculations using `perturbo.x`.

The input file for each calculation using `perturbo.x` is similiar to the silicon case (*"example02-silicon-perturbo"*, link). The main difference is the `'imsigma'` calculation, where users can use a variable called polar_split to specify whether they want to compute the full matrix element (polar plus non-polar part), or just the polar or nonpolar part.

- For the long-range (polar) part, we set `polar_split='polar'` in the input file. In this example, we use `'cauchy'` for \(\mathbf{q}\) point importance sampling and set the variable cauchy_scale for the Cauchy distribution.
- For the short-range (nonpolar) part, we use `rmpolar` for the variable polar_split and `'uniform'` for sampling.

Remember to converge both the long- and short-range parts of the e-ph matrix elements with respect to the number of \(\mathbf{q}\) points (the variable nsamples). If polar_split is not specified, `perturbo.x` will compute e-ph matrix elements including both the short- and long-range interactions, which typically has a slow convergence with respect to number of \(\mathbf{q}\) points.

Graphene: 2D Material

Directory: *example05-graphene-2d/* link

Run the preliminary calculations (scf, phonon, nscf, and Wannier90) in the directory *"pw-ph-wann"*. The input file for a 2D material for `qe2pert.x` requires to an extra variable, system_2d `= .true`. Here is the input file:

```
&qe2pert
  prefix='graphene'
  outdir='./tmp'
  phdir='../pw-ph-wann/phonon/References/save'
  nk1=36, nk2=36, nk3=1
  dft_band_min = 1
  dft_band_max = 11
  num_wann = 2
  lwannier = .true.
  system_2d = .true.
/
```

In the input files for `perturbo.x`, the user does not need to specify any variable related to 2D systems, since `perturbo.x` will know from the *'prefix"_epwan.h5*. When running the calculation modes `'setup'` or `'trans'`, the carrier concentration units are cm^{-2} instead of cm^{-3}. In this example, we focus only on the two bands that cross the Dirac cone of graphene. The band index is 1 for the valence and 2 for the conduction band. In the electron mobility calculation, we set accordingly both band_min and band_max to 2. In the hole mobility calculation, both band_min and band_max are set to 1.

Aluminum: Metal

🗀 **Directory:** *example06-aluminum/* link

Run all the preliminary calculations (scf, phonon, nscf, and Wannier90) in the *"pw-ph-wann"* directory. Run `qe2pert.x` to obtain the *'prefix'_epwan.h5* file. Run the desired calculations with `perturbo.x`. The input files are similar to those in *"examples/example01"* and *"examples/example02"*.

Automatic Generation of Input Files for Perturbo

The PERTURBO code has many calculation modes (specified by the calc_mode variable). Each calculation mode implies different mandatory and optional input parameters. In order to simplify and systematize the input files for the user, we provide the `generate_input.py` python script which generates the PERTURBO input files for different calculation modes.

To use the script, go to the *utils* directory of the PERTURBO code folder:

```
$ cd [perturbo_path]/utils
```

Suppose, we would like to generate the input file for the calculation mode `ephmat`. To do this, run:

```
$ ./generate_input.py --calc_mode ephmat
```

For a shorter version, one can specify `-c` instead of `--calc_mode`.

Then, the input file (called by default *pert.in*) is generated:

```
! This input file for PERTURBO was generated by generate_inpu
t.py script

&perturbo
! ***Mandatory parameters***
calc_mode = 'ephmat'
prefix = 'prefix'
fklist = 'prefix.kpt'
fqlist = 'prefix.qpt'


! ***Optional parameters***
! band_min = 1
! band_max = 9999999
! phfreq_cutoff = 1.0
/
```

It contains a block of mandatory parameters for this calculation mode and a block of optional ones, which is commented. As one can see, this input file containes some *typical* values for the input parameters. The user should modify them for a given calculation.

Setting the variables is also possible using the scipt. For example, to set `prefix` to `'si'` and `band_min` to `'10'`, run:

```
$ ./generate_input.py -c ephmat --prefix si --band_min 10
```

The values of these parameters were changed in the *pert.in* file. Note, that since we specified an optional parameter `band_min`, it was uncommented in the input file.

To change the name of the input file, run the script with `-i your_name.in` option. Setting the input parameter values from the scipt could be usefull in the case, when one needs to create automatically many different input files.

In order to generate the input files for the `qe2pert.x` calcuation, select `-c qe2pert`. Run the script with `-h` to get the whole list of possible options.

To get a *typical* input file without running the script, select the calculation type here: Select...

Relaxation Time from the 'imsigma' Calculation

Having computed the $\operatorname{Im}\Sigma$ values from the `'imsigma'` PERTURBO calculation (described here), one can find the relaxation time τ in the following way:

$$\tau = \frac{\hbar}{2} \frac{1}{\operatorname{Im}\Sigma}.$$
The scattering rate can be then found as the inverse of the relaxation time, τ^{-1}.

In order to calculate the relaxation times and the scattering rates from the `'imsigma'` calculation, we provide the `relaxation_time.py` Python script. To use it, you should have a *'preifx'*.imsigma file obtained as an output from the `'imsigma'` calculation.

Run the script in the directory where the *'preifx'.imsigma* file is located:

```
$ [perturbo_path]/utils/relaxation_time.py
```

If you have more than one *.imsigma* file in the directory, specify the file name with `--imsigma_file [file.imsigma]` (or `-i [file.imsigma]`) option.

The script generates the file called *relaxation_time.dat*, which has the following format:

```
 # it   ik  ibnd      E(ibnd)(eV)   Relaxation time(in fs)   Sc
attering rate (in THz)
   1       1     1      6.955370    2.6511488462206518e+01  3.7
719496641071323e+01
  ......
  ......
 #-------------------------------------------------------------
```

The first four columns are the same as in the *'prefix'.imsigma* file, which are: 1) the dummy variable for the temperature, 2) the number of $\mathbf{k}$ point, 3) the band number, 4) the energy. The 5th and 6th columns are the relaxation time (in fs) and the scattering rate (in THz).

Release Notes of PERTURBO 2.0

New features and improvements included in the 2.0 version of PERTURBO:

- Compatibility with QE 7.0
- Magnetotransport calculations
- Calculations on magnetic systems with collinear spin
- High-field transport
- Interface to TDEP for anharmonic phonons
- Bug fixes, performance improvements, test suite added, output format improved and more

Compatibility With Earlier Versions of Quantum Espresso

We recommend to use Perturbo with QE7.0. However, if using an earlier version of QE is necessary, a user needs to download PERTURBO 1.0 (the link will work only for users who are already added as collaborators to the GitHub project, more details here).

By default, the version 1.0 of PERTURBO will be compatible with QE6.5. For QE6.4, add the following flag in the *make.sys* file:

```
FFLAGS += -D__QE64
```

Changes of Transport Calculation Mode

The transport calculations mode names are changed.

- The non-magnetic RTA and ITA modes correspond to `calc_mode = 'trans-rta'` and `calc_mode = 'trans-ita'` respectively.
- Magnetic RTA and ITA correspond to `calc_mode = 'trans-mag-rta'` and `calc_mode = 'trans-mag-ita'`.

The boltz_nstep parameter is set to zero by default only for the non-magnetic RTA calculation and set to 50 by default for other transport calculation modes. The user can change the boltz_nstep value from the input file.

Output Formats

YAML output

All the data that outputs to ASCII text files, is now also written in YAML files (the text output being fully preserved). The YAML files can be easily postprocessed with Python using the PyYAML package.

Change of prefix_tdf.h5 format

The format HDF5 output file of the trans calculation mode (*prefix*_tdf.h5) was changed and made more userfriendly. Each line from the temper file now correspond to a separate group (called `configuration`). Inside each group there *might* be the `iterations` subgroup if the transport calculation was iterative.

Quantum Espresso to PERTURBO input parameters

Name	Type	Description
`prefix`	string	Job name prefix. It should be the same as the prefix used in QE. *Typical:* `prefix`
`outdir`	string	Name of the directory where the QE nscf output directory prefix.save is located, and where the e-ph matrix elements prefix_elph.h5 will be stored. *Typical:* `./tmp`
`phdir`	string	Name of the directory where the phonon "save" directory is located. *Typical:* `phdir`
`dft_band_min`	integer	Lowest band index used in Wannier90. *Default:* `1`
`dft_band_max`	integer	Highest band index used in Wannier90. Be default, it will be reset to the highest band index in the DFT results. *Default:* `10000`
`dis_win_min`	real	The 'dis_win_min' used in Wannier90, the lower boundary of the outer windows. *Default:* `-9999.0`
`num_wann`	integer	Number of Wannier functions. *Default:* `1`

`system_2d`	logical	Set it to .true. if the system is 2D. *Default:* `.false.`
`nk1`	integer	Number of k points along x-axis used in the Wannierization. *Typical:* `8`
`nk2`	integer	Number of k points along y-axis used in the Wannierization. *Typical:* `8`
`nk3`	integer	Number of k points along z-axis used in the Wannierization. *Typical:* `8`
`debug`	logical	Set to .true. to turn on the debug mode, in which the code stop after g(k,q) (does not compute g in wannier basis) *Default:* `.false.`
`lwannier`	logical	Set to .true. to rotate the wavefunctions using Wannier unitary matrix before computing e-ph matrix elements. *Default:* `.true.`
`load_ephmat`	logical	Set to .true. to load prefix_elph.h5 from the directory specified by the variable outdir. *Default:* `.false.`

`eig_corr`	string	File containing the electron eigenvalues on the (nk1, nk2, nk3) grid. The format of this file is the same as the file `prefix.eig` generated in Wannier90. if present, `qe2pert.x` will read the eigenvalues from this file, rather than Kohn-Sham eigenvalues from QE-nscf calculation. This is usually used when one wants to use modified eigenvalues (e.g., from GW). *Typical:* `eig_corr`
`polar_alpha`	real	Convergence parameter used in the Ewald sum when computing the polar correction in polar materials. The default value is 1.0. *Default:* `1.0`
`asr`	string	Indicates the type of Acoustic Sum Rule imposed. *Default:* `crystal` *Options:* [' `.false.` ', ' `simple` ', ' `crystal` ']
`thickness_2d`	real	Thickness of the 2d system, used in the 2D polar e-ph correction. Only needed when system_2d=.true. *Default:* `6.0`

PERTURBO input parameters

Job control		
Name	**Type**	**Description**
`prefix`	string	Job name prefix. It should be the same as the prefix used in QE. *Typical:* `prefix`
`calc_mode`	string	Calculation mode. *Options:* [' `bands` ', ' `phdisp` ', ' `ephmat` ', ' `setup` ', ' `imsigma` ', ' `meanfp` ', ' `trans` ', ' `trans-pp` ', ' `dynamics-run` ', ' `dynamics-pp` ']
`fklist`	string	Name of the file containing the k-point list (in crystal coordiates). *Typical:* `prefix_tet.kpt`
`fqlist`	string	Name of the file containing the q-point list (in crystal coordiates). *Typical:* `prefix_phdisp.qpt`
`ftemper`	string	Name of the file containing values for the temperature (K), chemical potential (eV), and carrier concentration (cm^{-2} or cm^{-3}). *Typical:* `prefix.temper`
`debug`	logical	Debug mode. *Default:* `.false.`
`hole`	logical	Set to .true. for calculations on hole carriers. *Default:* `.false.`

`tmp_dir`	string	The directory where the e-ph matrix elements are stored when `calc_mode='trans'` . *Typical:* `./tmp`
`load_scatter_eph`	logical	Read the e-ph matrix elements from the files in tmp_dir. Used for `calc_mode='trans'` . *Default:* `.false.`
`sampling`	string	Random q points sampling method. *Default:* `uniform` *Options:* [' `uniform` ', ' `cauchy` ']
`cauchy_scale`	real	Scale parameter gamma for the Cauchy distribution; used when `sampling='cauchy'` . *Typical:* `1.0`
`nsamples`	integer	Number of q-points for the summation over the q-points in imsigma calculation. *Default:* `100000`

Boltzmann Transport Equation

Name	Type	Description
`boltz_kdim`	integer	Number of **k** points along each dimension for the Boltzmann equation. *Default:* `(40,40,40)`
`boltz_qdim`	integer	Number of **q** points along each dimension for the Boltzmann equation. *Default:* `('boltz_kdim(1)','boltz_kdim(2)','boltz_kdim(3)')`

band_min	integer	Lowest band included. *Default:* 1
band_max	integer	Highest band included. *Default:* 9999999
boltz_emin	real	Bottom of the energy window for the Boltzmann equation. *Default:* -9999.0
boltz_emax	real	Top of the energy window for the Boltzmann equation. *Default:* 9999.0
boltz_nstep	integer	Number of iterations for solving the Boltzmann transport equation. *Typical:* 50
boltz_de	real	Energy step for the integrals in the Boltzmann equation. *Default:* 1.0
delta_smear	real	Smearing for the Dirac delta function. *Default:* 10.0
phfreq_cutoff	real	Phonon energy threshold. Phonons with energy smaller than phfreq_cutoff will be excluded. *Typical:* 1.0
trans_thr	real	Threshold for the iterative procedure. *Default:* 0.002
		Polar correction
Name	**Type**	**Description**

`polar_split`	string	Polar correction mode. *Default:* `''` *Options:* [" `''` ", ' `polar` ', ' `rmpol` ']

Ultra-fast dynamics

Name	Type	Description
`time_step`	real	Time step for the carrier dynamics. *Typical:* `1.0`
`output_nstep`	integer	Print out the results every `output_nstep` time steps. *Default:* `1`
`boltz_init_dist`	string	Initial electron distribution at time zero. *Typical:* `gaussian` *Options:* [' `restart` ', ' `lorentz` ', ' `fermi` ', ' `gaussian` ']
`boltz_init_e0`	real	Energy parameter used to generate initial distribution. Needs to be specified for `boltz_init_dist='lorentz'` (center), `'gaussian'` (center), or `'fermi'` (chemical potential). *Typical:* `1.0`
`boltz_init_smear`	real	The broadening or width of the initial distribution for `boltz_init_dist='lorentz'` or `'gaussian'` , or temperature (in meV) for `'fermi'` . *Typical:* `1.0`
`solver`	string	Solver type for the Boltzmann transport equation. *Default:* `rk4` *Options:* [' `euler` ', ' `rk4` ']