Perturbo

version 1.0

Last generated: January 24, 2020

Caltech

Table of Contents

Getting started

Tutorials

Other examples

Release Notes

Utils

Input parameters

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 other 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). 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 structure of PERTURBO, we refer the users to the manuscript accompying the source code:

- Jin-Jian Zhou, Jinsoo Park, I-Te Lu, Marco Bernardi, Perturbo: a software package for electron-phonon interactions, charge transport and ultrafast dynamics, arXiv:xxxx.xxxx (2020).

When using results from PERTURBO in your publications, please cite the above paper and acknowledge the use of Perturbo.

Features

PERTURBO has the following stable features:

- Phonon-limited carrier mobility, electrical conductivity and Seebeck coefficient
- Phonon-limited carrier mean free path and relaxation times
- Imaginary part of e-ph self-energy and e-ph scattering rates
- e-ph matrix elements for nonpolar and polar materials, and their Wannier interpolation
- Interpolated electronic band structure and phonon dispersion
- Ultrafast carrier dynamics with fixed phonon occupation

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

Installation and compilation

Note: PERTURBO uses a small number of subroutines from the PWSCF and Phonon packages of QE. Therefore, it needs to be compiled on top of QE. We assume that the users have already compiled QE successfully.

Download

In order to download the source code, contact us (page 4) and we will:

- [Recommended] add you as a collaborator in our GitHub project
- [If you do not have a GitHub account] send you a *.tar.gz* file.

Clone from GitHub (or extract *.tar.gz*) into the QE directory. There are four 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`
- *"examples"* has input files for examples and tutorials on `perturbo.x` and `qe2pert.x`

Compilation

There are two files in the *"perturbo"* directory, *Makefile* and *make.sys*. Modify *make.sys* to make it suitable for your system or copy an existing *make.sys.XXX* file from the directory *"config"*.

```
$ vim make.sys
or
$ cp ./config/make.sys.XXX ./make.sys
```

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

```
$ make
```

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

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 .

Organization of the tutorials

In this section, we will go through two tutorial examples:

- *example01-silicon-qe2pert*: to learn how to use `qe2pert.x` to generate a '*prefix*'*_epwan.h5* file
- *example02-silicon-perturbo*: to learn how to use `perturbo.x` to run perturbo calculations

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.

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 and 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 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. In this step, make sure that the number of points is commensurate with the number of points used in the nscf and Wannierization calculations. For example, a grid of 8x8x8 can be used with a wannierization 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 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 points in the phonon calculation.

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

In our silicon example, the total number of points is 29. It is fine to forgo the previous step and obtain the number of 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 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 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 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 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 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 , 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 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 point (determined by the nscf step) and 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 (page 0): needs to be the same as the prefix used in the input files for QE.
- outdir (page 0): contains the save directroy obtained from the nscf calculations. The calculated e-ph matrix elements will be stored in this directory.
- phdir (page 0): is the save directory inside which we collected all the phonon information.
- nk1 (page 0), nk2 (page 0), nk3 (page 0): are the number of points along each direction used in the nscf and Wannier90 calculations.
- dft_band_min (page 0) and dft_band_max (page 0): 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 (page 0): the number of Wannier functions.
- lwannier (page 0): 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 (page 0): 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 (page 0): if the materials is two-dimensional, so that in one direction only one 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 comsuption 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 here (page 0)).

PERTURBO calculation

In this section, we will discuss the features (or calculation modes) of `perturbo.x`. The variable for the calculation mode is `calc_mode`. Here are the optional values for `calc_mode`, 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 relaxition 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.

In the following, we use silicon as an example to demonstrate the features of PERTURBO (see the directory *"example02-silicon-perturbo/perturbo"*, link). **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 section qe2pert.x (page 0). The file *si_epwan.h5* is inside the directory *"example02-silicon-perturbo/qe2pert.x"*, link. For each calculation mode, we also provide reference results in the directory *"References"*. In all calculations, the same prefix value as in the QE DFT calculation should be used.

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 (page 0): the same prefix used in **'prefix'_epwan.h5**
- calc_mode (page 0): set to `'bands'`
- fklist (page 0): 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 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 , , and 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 point list, rather than specifying the path, by providing the number of points in the first line, the coordinates of each 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
```

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 , , and coordinates of the crystal momenta **in crystal coordinates**. The 5th column is the energy, in eV units, of each electronic state.

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 (page 0): the same prefix used in **'prefix'_epwan.h5**
- calc_mode (page 0): set to 'phdisp'
- fqlist (page 0): 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 (page 0) specified in `fklist` (in the previous section (page 0)).

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 , , and coordinates of the crystal momenta, in crystal coordinate. The 5th column is the phonon energy in meV units.

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 and points. In a typical scenario, one computes the e-ph matrix elements for a chosen point as a function of point

Requires to specify at least 7 variables:

- prefix (page 0): the same prefix as in *‘prefix’_epwan.h5*
- calc_mode (page 0): set to ‘ephmat’
- fklist (page 0): the file containing a list of points (see the section (page 0) on `calc_mode='bands'`)
- fqlist (page 0): the file containing a list of points (see the section (page 0) on `calc\_mode='bands'`)
- band_min (page 0), band_max (page 0): bands used for the band summation in computing e-ph matrix elements
- phfreq_cutoff (page 0): 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. –NOT TRUE, NEED PATCHING**

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 (page 0) and band_max (page 0).

The variable phfreq_cutoff (page 0) 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 (page 0) to a large value, otherwise too many phonon modes will be excluded from the calculations.

For the format of fklist (page 0) or fqlist (page 0) files, please refer to the section (page 0) 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 (page 0) to band_max (page 0). 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 point. The 2nd column is the point coordinate used for plotting. The 3rd and 4th columns are the dummy index and the 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.

calc_mode = 'setup'

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

Computes: Set up transport property calculations (i.e., electrical conductivity, carrier mobility and Seebeck) by providing points, point tetrahedra and (if needed) finding chemical potentials for given carrier concentrations

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

- prefix (page 0): same prefix as in *'prefix'_epwan.h5*
- calc_mode (page 0): set to 'setup'
- hole (page 0): 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 (page 0): number of points along each dimension of a point grid for the electrons momentum. This Gamma-centered Monkhorst-Pack point grid is employed to compute the mobility or conductivity.
- boltz_qdim (page 0): number of 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

grid, no need to specify these variables. Only phonons with mmentum on the grid are considered in the calculations of e-ph scattering.

- boltz_emin (page 0), boltz_emax (page 0): energy window (in eV units) used to compute transport properties. The suggested values are from 6 k_BT below E_F (boltz_emin (page 0)) to 6k_BT above E_F (boltz_emax (page 0)), where E_F is the Fermi energy, k_B the Boltzmann constant, and T is temperature in K units.
- band_min (page 0), band_max (page 0): band window for transport property calculations
- ftemper (page 0): 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 grid of 80 x 80 x 80 for electrons, which corresponds to boltz_kdim(i)=80, and use a grid for phonons of the same dimension as the grid. When a phonon grid different from the electron grid is desired, the user need to provide the 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 find all the relevant points (both irreducible and reducible points) and the tetrahedron needed for BZ integration for the given energy window and band windowm compute the DOS at the given energy window. It also compute carrier concentrations at given chemical potentials or determine 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 (page 0) 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 and 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 points (both in the irreducible wedge and full grid) and the associated 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 points in the energy window of interest. Note that the irreducible 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 (page 0)) or `'imsigma'` (below (page 0)).
- *'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.

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 for states in a range of bands and with crystal momenta read from a list. The scattering rates can also be obtained using

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

- prefix (page 0): the same prefix as the file *'prefix'_epwan.h5*
- calc_mode (page 0): set to `'imsigma'`
- band_min (page 0), band_max (page 0): bands used for transport property calculations
- ftemper (page 0): the filename of a file containing temperature, chemical potential, and carrier concentration values (see the format (page 0))
- fklist (page 0): the filename of a file containing the coordinates of a given electron point list (see the format (page 0))
- phfreq_cutoff (page 0): 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 (page 0): the broadening (in meV) used for the Gaussian function used to model the Dirac delta function
- fqlist (page 0): the filename of a file containing the coordinates of a given phonon point list will be used to compute the e-ph self-energy. For the format, see the section (page 0) on the calculation mode `'bands'`. This is optional. If `fqlist` is absent or `fqlist_=''`, random points will be generated (see below).
- sampling (page 0): sampling method for random points used in e-ph self-energy calculation. The default value is `'uniform'`, indicates sampling random points in the first BZ following uniform distribution. Another option is `'cauchy'`, sampling random points following Cauchy distribution, which is useful for polar materials. Note that random points from other importance sampling methods or points on regular MP grid is also possible, one just needs to pre-generate the points list to a file, and

pass the file to `perturbo.x` via `fqlist`.

- cauchy_scale (page 0): the width of the Cauchy function; used only when sampling (page 0) is 'cauchy'.
- nsamples (page 0): number of random points sampled to compute the imaginary part of the e-ph self-energy for each 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 points in the *fklist* file (in this case, we use the irreducible Monkhorst-Pack 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 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 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-comsuming 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 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 points and the smearing parameter (delta_smear (page 0)). 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 (it's convenient to use 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 (page 0)). 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.

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 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 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 .

calc_mode = 'trans'

The calculation mode `'trans'` computes the electrical conductivity and carrier mobility tensors. The code can compute these quantities using the relaxation time approximantion (RTA) of the Boltzmann transport equation (BTE) or an iterative approach (ITA) to fully solve the linearized BTE.

Relaxation time approximation (RTA)

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

Computes: The phonon-limited conductivity and carrier mobility using the RTA of the BTE

Note: The user needs to run the calculation modes `'setup'` and then `'imsigma'` since this calculation mode relies on their outputs

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

- calc_mode (page 0): set to `'trans'`

- boltz_nstep (page 0): set to 0, which means computing the mobility using the RTA

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

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

 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  ! RTA
/
```

Before running perturbo.x, remember to put the following files in the current 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 three output files:

- *'prefix'.cond* contains the conductivity and mobility tensors as a function of temperature
- *'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 for each temperature in HDF5 format

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    0.256282E+05  -0.867734E-0
6   0.256282E+05  -0.643904E-06  -0.266436E-04   0.256282E+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    0.160830E+04  -0.544546E-0
7   0.160830E+04  -0.404082E-07  -0.167202E-05   0.160830E+04
```

The calculated electron mobility at 300 K is ~ 1608 $cm^2V^{-1}s^{-1}$, in reasonably good agreement with the experimental value of roughly 1400 $cm^2V^{-1}s^{-1}$.

(THE VALUE DEVIATES SIGNIFICANTLY FROM WHAT I(JINSOO)'VE DONE BEFORE. WHY IS THIS? WE SHOULD RE-DO THE CALCULATIONS)

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


   ......
   6.636612    1.7531179753260076E+01    0.316717E-02   0.17876
2E-07   0.316728E-02   0.936616E-08   0.153835E-06   0.316718
E-02
   ......
   ......
```

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 column 3-8 is the TDF 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 in the file. In this case, we look at one temperature and one concentration, so there is only one block in the file.

In more rigorous calculations, the user will need to converge the conductivity and mobility with respect to the number of and points, namely the variables boltz_kdim (page 0) and boltz_qdim (page 0).

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

Iterative approach (ITA)

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

Computes: The phonon-limited conductivity and carrier mobility using ITA

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 (page 0): is set to `'trans'`
- boltz_nstep (page 0): contains the maximum number of iterations in the iterative scheme for solving Boltzmann equation, where a typical value is 10
- phfreq_cutoff (page 0): contains phonon threshold (meV). Phonons with energy smaller than the cutoff will be ignored.
- delta_smear (page 0): contains broadening (meV) for a Gaussian function to present the Dirac delta function
- tmp_dir (page 0): contains output directory containing the e-ph matrix elements used in the calculations
- load_scatter_eph (page 0): 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'

 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 current directory (*“pert-trans-ITA-electron”*):

- *‘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-comsuming 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 3 output files, *‘prefix’.cond*, *‘prefix’.tdf*, and *‘prefix’_tdf.h5*, 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 .

calc_mode = 'trans-pp'

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

> **Computes:** Seebeck coefficient. Note that phonon drag effects are not included in this calculation.

Uses the same input file as the `'trans'` calculation mode, but requires the additional file *'prefix'_tdf.h5* obtained in the `'trans'` calculation. The Seebeck calculation is a quick post-processing of the 'trans' calculation, which needs to be done before running `'trans-pp'`.

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 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   0.158023E+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   0.425885E+00   0.186328E-0
6   0.425883E+00  -0.791938E-07  -0.329487E-06   0.425885E+00
```

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”*, link .

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

For the ultrafast dynamics, one needs first to perform the `calc_mode = 'setup'` calculation, which is described here (page 0). So, we assume that the user has already performed the `'setup'` calculation. From the `'setup'` calculation, we retain the following files necessary for the dynamics calculations: *'prefix'.temper* and *'prefix'_tet.h5*.

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

- preifx (page 0): the same prefix as in *'prefix'_epwan.h5*
- calc_mode (page 0): set to `'dynamics-run'`
- boltz_kdim (page 0): grid for electrons, here we use a 80x80x80 grid
- boltz_qdim (page 0): grid for phonons, specify if it is different from the grid, here we use the same grid as grid
- boltz_emin (page 0), boltz_emax (page 0): energy window (in eV units), use the same as in the `'setup'` calculation
- band_min (page 0), band_max (page 0): band range
- ftemper (page 0): the filename of a file containing temperature, chemical potential, and carrier concentration values (see the format (page 0))
- time_step (page 0): simulation time step, set to its typical value, 1 fs
- boltz_nstep (page 0): total number of time steps, set to 50; here we perform a relatively short simulation of 50 fs
- output_nstep (page 0): an optional variable to shorten the output; the output time step is determined in the following way: , where is the simulation time step
- solver (page 0): BTE solver type, set to `'euler'`, here we use the Euler first order solver of BTE
- boltz_init_dist (page 0): 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 (page 0): in this example, the Gaussian distribution is centered around 7.4 eV
- boltz_init_smear (page 0): we select a 40 meV smearing
- phfreq_cutoff (page 0): we select a 1 meV phonon energy cutoff
- delta_smear (page 0): 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 = 6.9
 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 = 7.4 ! 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 (page 0) 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
- `dynamics_run_[i]` : an HDF5 group that contains information about the i^{th} 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 (page 0)), can be different for different dynamics_run_[i]
    - `snap_t_0` :
    - `snap_t_[j]` : the distribution function for time : , where is the output time step. Each column of the array corresponds to the band index
    - `snap_t_[num_steps]` :
    - `time_step_fs` : the output time step (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 pakage python package h5py . As an example, we present here a simple Python script that visualizes the distribution function for the time 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.

calc_mode = 'dynamics-pp'

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

Computes: Postprocessing of the ultrafast dynamics calculations

In this section we aim to calculate the Brillouin zone-averaged *energy*-dependent carrier population . Having calculated the distribution function , one can find in the following way:

The integral of over the energy gives the number of carriers per unit cell as a function of time.

In order to calculate the quantity, one needs to have all the files required for the `calc_mode='dynamics-run'` calculation (previous section (page 0)) 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 (page 39), 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
```

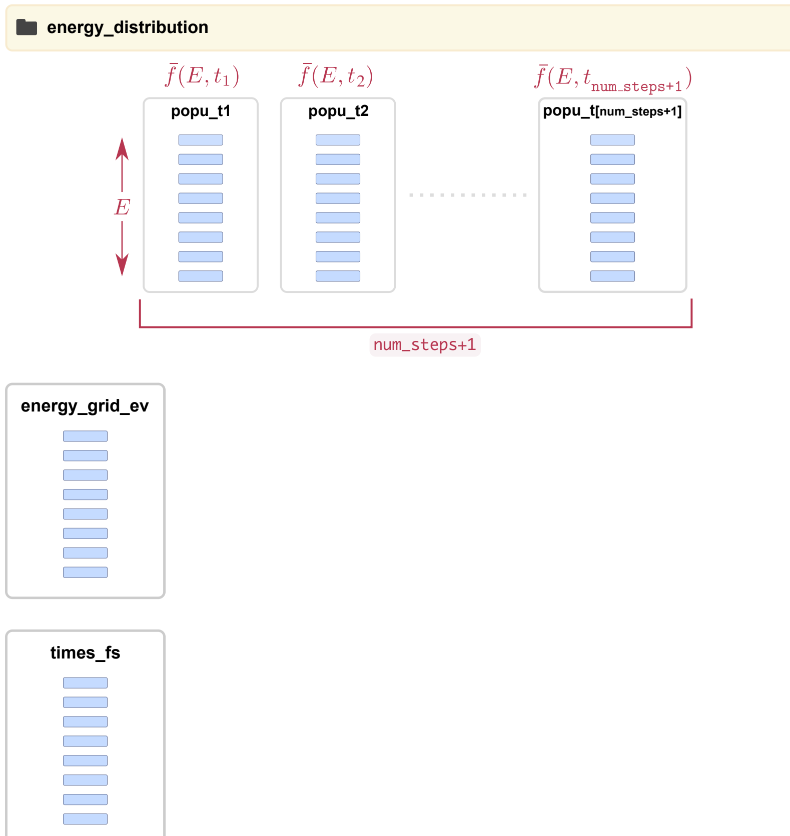
On the output, we obtain the following files:

- *si_popu.h5*: an HDF5 file that contains all the necessary information for
- *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`:
  - `popu_t[j]`: the carrier population at time
  - `popu_t[num_steps+1]`:
- `energy_grid_ev`: the grid of energies in eV; the number of energy grid points is given by
- `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 , run:

```
#!/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()
```

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 (with 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 (page 0) variable) and DFT bands (dft_band_min (page 0) and dft_band_max (page 0)) 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 (page 0) and dft_band_max (page 0). 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 (page 0) 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 point importance sampling (page 0) and set the variable cauchy_scale (page 0) for the Cauchy distribution.
- For the short-range (nonpolar) part, we use `rmpolar` for the variable polar_split (page 0) and `'uniform'` for sampling (page 0).

Remember to converge both the long- and short-range parts of the e-ph matrix elements with respect to the number of points (the variable nsamples (page 0)). If polar_split (page 0) 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 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 (page 0) `= .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 (page 0) and band_max (page 0) to 2. In the hole mobility calculation, both band_min (page 0) and band_max (page 0) 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"*.

Release Notes 1.0

We are pleased to announce the first release of the PERTURBO code.

Automatic generation of exemplified input files for PERTURBO

The PERTURBO huicode has many calculation modes (specified by the calc_mode (page 0) 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 a python script which can generate the PERTURBO input files for different calculation modes.

To use the script, go to the *utils* directory:

```
$ cd ./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 see a *typical* input file without running the script, select the calculation type here: Select...

Relaxation time from the 'imsigma' calculation

Having computed the values from the `'imsigma'` PERTURBO calculation (described here (page 0)), one can find the relaxation time in the following way:

The scattering rate can be then found as the inverse of the relaxation time, .

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:

```
$ [your_prerturbo_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 (page 0), which are: 1) the dummy variable for the temperature, 2) the number of 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).

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. *Default:* `10000`
`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 (in Å), 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 coordiate). *Typical:* `prefix.kpt`
`fqlist`	string	Name of the file containing the q-point list (in crystal coordiate). *Typical:* `prefix.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 (page 0)='trans'` . *Typical:* `./tmp`
`load_scatter_eph`	logical	Read the e-ph matrix elements from the files in tmp_dir. Used for `calc_mode (page 0)='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 (page 0)='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:* `(1,1,1)`
`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 (in eV) for the Boltzmann equation. *Default:* `-9999.0`
`boltz_emax`	real	Top of the energy window (in eV) 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 (in meV) for the integrals in the Boltzmann equation. *Default:* `1.0`
`delta_smear`	real	Smearing (in meV) for the Dirac delta function. *Default:* `10.0`
`phfreq_cutoff`	real	Phonon energy threshold (meV). 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 ', ' rmpolar ']

Ultra-fast dynamics

Name	Type	Description
time_step	real	Time step for the carrier dynamics (in fs). *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:* `restart` *Options:* [' restart ', ' lorentz ', ' fermi ', ' gaussian ']
boltz_init_e0	real	Energy (in eV) parameter used to generate initial distribution. Needs to be specified for `boltz_init_dist (page 0)='lorentz'` (center), `'gaussian'` (center), or `'fermi'` (chemical potential). *Typical:* `1.0`
boltz_init_smear	real	The broadening or width (in meV) of the initial distribution for `boltz_init_dist (page 0)='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` ']