



Perturbo

version 1.0

Last generated: January 01, 2020

Caltech

© 2020 Caltech. This is a boilerplate copyright statement... All rights reserved. No part of this publication may be reproduced, distributed, or transmitted in any form or by any means, including photocopying, recording, or other electronic or mechanical methods, without the prior written permission of the publisher, except in the case of brief quotations embodied in critical reviews and certain other noncommercial uses permitted by copyright law.

Table of Contents

Getting started

Introduction	2
Supported features	3
Installation and compilation	4
Research team	6

Release Notes

1.0 Release notes	7
-------------------------	---

Workflow

qe2pert.x	8
perturbo.x	14

Tutorials

Introduction	39
Silicon	40
GaAs	41
Graphene	42
Aluminum	43

Input parameters

qe2pert.x	44
perturbo.x	46

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 languages (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 a 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 a HDF5 output format for PERTURBO to read.

For more details on the code structure of PERTURBO, we refer the users to the manuscript accompanying 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: (TO BE MODIFIED)

- Phonon-limited carrier mobility and electrical conductivity
- 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

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 know how to compile QE successfully

Download

Change into the QE directory – and download PERTURBO from Github: (TO BE MODIFIED)

```
$ cd <Quantum Espresso directory>
$ git clone https://github.com/jinjianzhou/perturbo.git
```

Once PERTURBO has been downloaded, change into the directory “perturbo”. (UNTAR?)

```
$ cd perturbo
```

There are four subdirectories inside the directory “perturbo”:

- “configure” contains the system-dependent makefiles *make.sys.XXX*
- “pert-src” contains the source code to compute electron dynamics
- “qe2pert-src” contains the source to convert the output from QE to the format read by PERTURBO.
- (MORE ABOUT QE2PERT)
- “examples” has input files for examples and tutorials on *perturbo.x* and *qe2pert.x*

Compilation

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 “configure”.

```
$ 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, you are invited to visit the [group website](#).

Release Notes 1.0

Version 1.0 is the first release of the PERTURBO code.

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, the first step is to use `qe2pert.x` to compute the e-ph matrix elements on a coarse k- and q-point Brillouin zone grid, 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 “reference”.

Step 1: scf calculation

Directory: examples/example02-silicon-qe2pert/pw-ph-wan/scf

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: examples/example02-silicon-qe2pert/pw-ph-wan/phonon

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 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. In this step, make sure that the number of q-points is commensurate with the number of k-points used in the nscf and

Wannierization calculations. For example, a q-grid of 8x8x8 can be used with a wannierization 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 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 q-points in the phonon calculation.

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

In our silicon example, the total number of q-points is 29. It is fine to forgo the previous step and obtain the number of 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 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 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 q-points before running the phonon calculation. Note however that this step is optional.

Step 3: nscf calculation

Directory: examples/example02-silicon-qe2pert/pw-ph-wan/nscf

We now run the nscf calculations needed to generate the wavefunctions on the full 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 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: examples/example02-silicon-qe2pert/pw-ph-wan/wann

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: examples/example02-silicon-qe2pert/qe2pert

We are ready to compute the e-ph matrix elements on the coarse k-point (determined by the nscf step) and 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 for the executable qe2pert.x:

```

&qe2pert
prefix='si'
outdir='./tmp'
phdir='../pw-ph-wann/phonon/save'
nk1=8, nk2=8, nk3=8
qe_band_min = 1
qe_band_max = 16
num_wann = 8
lwannier=.true.
/

```

- *prefix* needs to be the same as the prefix used in the input files for QE.
- *outdir* contains the save directory 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 k-points along each direction used in the nscf and Wannier90 calculations.
- *qe_band_min* and *qe_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 *qe_band_min*=5 and *qe_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. Set it to `.true.` to reduce computational cost.

- `system_2d`: if the material is two-dimensional, so that in one direction only one 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=1
$ mpirun -n 1 qe2pert.x -npools 1 -i qe2pert.in > qe2pert.out
```

The executables `perturbo.x` and `qe2pert.x` employ hybrid parallelization (MPI plus OpenMP). To speed up the calculations, the user can increase the number of OpenMP threads and MPI processes.

Note that the number of pools (`-npools`) has to be equal to the number of MPI processes (`-np` or `-n`), otherwise the code will stop. This task takes 8 hours on a single core, but it can be made much faster (minutes) using MPI plus OpenMP. We particularly recommend using threads with OpenMP (usually, setting `OMP_NUM_THREADS=` half the value of cores per node is a good choice). (**Jin-Jian, can you revise these statements? **). Once the calculation has completed, we obtain the output file `si_epwan.h5`, which is an HDF5 database with all the information needed to run `perturbo.x`.


PERTURBO calculation


In this section, we will discuss all the features (or calculation modes) of `perturbo.x`. The variable for the calculation mode is `calc_mode`. Here are the possible values for `calc_mode`, and the corresponding tasks carried out by PERTURBO:

- ‘bands’: interpolate electronic band structures using maximally localized Wannier functions
- ‘phdips’: interpolate phonon dispersion by Fourier transforming real-space interatomic force constants
- ‘ephmat’: interpolate e-ph matrix elements using maximally localized Wannier functions
- ‘setup’: set up transport calculations
- ‘imsigma’: compute the e-ph self-energy for electronic crystal momenta read from a list
- ‘meanfp’: compute the e-ph mean free path
- ‘trans’: compute electrical conductivity for metals, semiconductors, and insulators, or carrier mobility for semiconductors (computationally demanding for the iterative approach)
- ‘trans-pp’: compute the Seebeck coefficient
- (ADD ULTRAFAST DYNAMICS)

In the following, we use silicon as an example to demonstrate the features of PERTURBO (see the directory “examples/example01-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 below. The file `si_epwan.h5` is inside the directory “examples/example01-silicon-perturbo/qe2pert.x”. 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:** examples/example01-silicon-perturbo/perturbo/pert-band

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

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

- *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 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 units**. The last column is the number of points from the current crystal momentum to the next crystal momentum. One can also provide an explicit k-point list, rather than specifying the path, by providing the number of k points in the first line, the coordinates of each 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" (You may choose to copy the HDF5 file using `$ cp ../../qe2pert/si_epwan.h5 .`, or link the file using `$ ln -sf ../../qe2pert/si_epwan.h5 .`. We recommend linking rather than copying the HDF5 file due to the relatively large file size).

Run *pertubo.x*:

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

Remember to set the number of pools (-npools) equal to the number of MPI process (-n or -np). 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	-0.8385755999
.....				
3.7802390	0.00000	0.00000	0.00000	6.1762484317
0.0000000	0.50000	0.50000	0.50000	4.9707614733
.....				
3.7802390	0.00000	0.00000	0.00000	6.1762484335
0.0000000	0.50000	0.50000	0.50000	4.9707614740
.....				
3.7802390	0.00000	0.00000	0.00000	6.1762484335
0.0000000	0.50000	0.50000	0.50000	7.6304859491
.....				
3.7802390	0.00000	0.00000	0.00000	8.6898783415
0.0000000	0.50000	0.50000	0.50000	9.4530710727
.....				
3.7802390	0.00000	0.00000	0.00000	8.6898783428
0.0000000	0.50000	0.50000	0.50000	9.4530710753
.....				
3.7802390	0.00000	0.00000	0.00000	8.6898783429
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 units**. The 5th column is the energy, in eV units, of each electronic state.

`calc_mode = 'phdisp'`

Directory: examples/example01-silicon-perturbo/perturbo/pert-phdisp

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 *fklist* (see the section on `calc_mode='bands'`).

Remember to link (or copy) `si_epwan.h5` in the current directory using `ln -sf`

....

Run `perturbo.x`:

```
$ mpirun -n 1 <perturbo_bin>/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	12.9198400723
.....				
3.7802390	0.00000	0.00000	0.00000	-0.0000022344
0.0000000	0.50000	0.50000	0.50000	45.6922098051
.....				
3.7802390	0.00000	0.00000	0.00000	0.0000014170
0.0000000	0.50000	0.50000	0.50000	50.7258504163
.....				
3.7802390	0.00000	0.00000	0.00000	63.1558948005
0.0000000	0.50000	0.50000	0.50000	60.2661349902
.....				
3.7802390	0.00000	0.00000	0.00000	63.1558948005

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 units. The 5th column is the phonon energy in meV units.

calc_mode = 'ephmat'

Directory: examples/example01-silicon-perturbo/perturbo/pert-ephmat

Computes: The absolute values of the e-ph matrix elements, summed over the number of electronic bands, given two lists of k and q points. In a typical scenario, one computes the e-ph matrix elements for a chosen k-point as a function of 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 k points (see the section on calc_mode='bands')

- *fqlist*: the file containing a list of q points (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. -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 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 stabilities in the phonon calculations, and we recommend using a value between 0.5 to 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 in the calculations.

For the format of *fklist* or *fqlist*, 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 pertubo.x:


```
$ mpirun -n 1 <perturbo_bin>/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 have si.ephmat, which is shown next:

#	ik	xk	iq	xq	imod	omega(meV)	deform.
	pot. (eV/A)		g (meV)				
	1	0.00000	1	0.00000	001	12.919840	0.21992730
	8382E+00	0.118026594146E+02					
	1	0.00000	1	0.00000	002	12.919840	0.21992730
	8382E+00	0.118026594146E+02					
	1	0.00000	1	0.00000	003	45.692210	0.62122121
	7159E+01	0.177277853210E+03					
	1	0.00000	1	0.00000	004	50.725850	0.47205046
	0536E+01	0.127850679974E+03					
	1	0.00000	1	0.00000	005	60.266135	0.58888638
	0766E+01	0.146326884271E+03					
	1	0.00000	1	0.00000	006	60.266135	0.58888638
	0766E+01	0.146326884271E+03					
	1	0.00000	2	0.01698	001	12.919176	0.22218363
	1468E+00	0.119240540866E+02					
	1	0.00000	2	0.01698	002	12.919176	0.22218363
	1468E+00	0.119240540866E+02					
	1	0.00000	2	0.01698	003	45.618607	0.61808553
	2274E+01	0.176525258121E+03					
	1	0.00000	2	0.01698	004	50.785889	0.47547090
	2657E+01	0.128700934013E+03					
	1	0.00000	2	0.01698	005	60.266760	0.58943228
	2022E+01	0.146461770958E+03					
	1	0.00000	2	0.01698	006	60.266760	0.58943228
	2022E+01	0.146461770958E+03					
						

The 1st column is a dummy index for the k-point. The 2nd column is the k-point coordinate used for plotting. The 3rd and 4th columns are the dummy index and the 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.

`calc_mode = 'setup'`

 **Directory:** examples/example01-silicon-perturbo/perturbo/pert-setup-electron (pert-setup-hole)

 **Computes:** Set up transport property calculations (i.e., electrical conductivity, carrier mobility and Seebeck) by providing k-points, k-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*: same prefix as in 'prefix'_epwan.h5
- *calc_mode*: set to 'setup'
- *hole*: compute hole carrier concentration. By default *hole* is set to .false., and electron carrier concentration will be computed. Set it to .true. when one needs hole carrier concentration.
- *boltz_kdim(1)*, *boltz_kdim(2)*, *boltz_kdim(3)*: number of k points along each dimension of a k-point grid for the electrons momentum. This k-point grid is employed to compute the mobility or conductivity.
- *boltz_qdim(1)*, *boltz_qdim(2)*, *boltz_qdim(3)*: number of 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 k grid, no need to specify these variables.
- *boltz_emin*, *boltz_emax*: energy window (in eV units) used to compute transport properties. The suggested values are from 6 $k_B T$ below E_F (*boltz_emin*) to 6 $k_B T$ 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*: bands used for transport property calculations
- *femper*: the filename of a file containing the temperature(s), chemical

potential(s), and corresponding carrier concentration(s) for transport property calculations.

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

femper = 'si.temper'
/
```

In the input file pert.in, we use a k-grid of 80 x 80 x 80 for electrons, which corresponds to `boltz_kdim(i)=80`, and use a q-grid for phonons. When a phonon q-grid different from the electron k-grid is desired, the user has to provide the 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 electron mobility, 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`).

In this case, the femper 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 settings at which we want to compute the mobility. The logical variable in the first line can be set to 'T' (for true), as is the case here, or 'F' (for false). perturbo.x will determine the chemical potential using the carrier concentration and temperature provided as input in each of the rows below the first. If the logical variable is false ('F'), perturbo.x will use the chemical potential provided in each of the rows below the first. 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). There are two possible scenarios. If the user wants `perturbo.x` to determine the chemical potential corresponding to a given carrier concentration and temperature, the logical variable in the first line is 'T' and upon running `perturbo.x` `femper` file will be overwritten. In the `femper` file generated by `perturbo.x`, the logical variable will become 'F', and the chemical potentials in each line below the first will be replaced by the values found by `perturbo.x`, which can solve for the chemical potential in the energy window between `boltz_emin` and `boltz_emax` (so make sure this window is large enough to include the chemical potential). In a second scenario, the logical variable is 'F', and `femper` will not be overwritten and will be used as is. In short, if the users want to find the chemical potentials for each combination of carrier concentration and a temperature provided as input, they need to first set the logical variable as 'T' and run the code to generate the `femper` with the desired chemical potential to be used in the transport calculation.

Run `perturbo.x` with the following command (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
```

The calculation will take a few minutes or longer, depending the number of k- and 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 temperature 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 k-points (in the irreducible wedge) 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 k-points, and the associated k-point tetrahedra, in the energy window of interest. The format is different from that of *fklist* discussed above in the calculation mode 'bands'.
- *prefix.dos* contains the density of states (number of states per eV per unit cell volume) 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 `femper`, a new `femper` file is generated as output: that the `femper` 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 "examples/example01-silicon-perturbo/perturbo/pert-setup/pert-setup-hole". 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: examples/example01-silicon-perturbo/perturbo/pert-imsigma-electron (pert-imsigma-hole)

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 k read from a list. The scattering rates can also be obtained using $2/\hbar * \text{ImSigma}$

Specify up to 12 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
- *femper*: the filename of a file containing temperature, chemical potential, and carrier concentration values (for the format, see the section above on the calculation mode 'setup')
- *fklist*: the filename of a file containing the coordinates of a given electron k-point list (for the format, see the section on the calculation mode 'bands')
- *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
- *sampling*: sampling type, 'uniform' or 'cauchy'; the former is a uniform random Brillouin zone grid, and is the default, while the 'Cauchy' is a useful option for polar materials
- *cauchy_scale*: the width of the Cauchy function; used only when *sampling* is 'cauchy' for polar materials
- *nsamples*: number of q-points used to compute the imaginary part of the e-ph self-energy for each k-point
- *polar_split*: describes how to compute the e-ph matrix elements for in the presence of a polar e-ph interaction (here, the ab initio Frohlich interaction). Three options: "(none)", 'polar', or 'rmpol'. Leave it blank if users want to describes how to compute the entire e-ph matrix elements (including the polar contribution). Set it to 'polar' when computing only for elements (including the polar contribution); set if to 'rmpol' when computing only the nonpolar part of the matrix elements (the total minus the polar part). This variable is relevant only for polar materials.

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 a k-points in the fklist file (in this case, we use the irreducible Monkhorst-Pack 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 k-point path in the fklist file instead. The temperature, chemical potential, and carrier concentration values

for computing the e-ph self-energy are given in the `femper` file, `si.temper`, obtained from the `perturbo` 'setup' process. 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 `femper`.

Here we use a uniform random sampling (`sampling='uniform'`) with 1 million randomly sampled 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`). In the imaginary part of the e-ph self-energy (or equivalently, e-ph scattering rate) calculations, the energy difference between the initial and final states must match the energy of the involved phonon, within a range of $\pm 3 \cdot \text{delta_smear}$ from gaussian smearing. To converge the e-ph self-energy, one must vary the input variables 'nsample' and 'delta_smear'. A small parameter $\eta \approx 5$ meV usually a good starting point, as this value is known to sufficiently converge the scattering rates in GaAs (Reference Jinjian).

Before running `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
```

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 (one separate file for each phonon mode, 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 Approximatio
n #
#      (WARNING: only output a part of the eigenstate
s)      #

#=====
#
#NO.k:    450    NO.bands:    2    NO.T:    1    NO.modes:    1
# it      ik    ibnd      E(ibnd)(eV)      Im(Sigma)(meV)
# Temperature(T)= 25.85203 meV; Chem.Pot.(mu)= 6.55048 eV
#   1      1      1      6.955370      1.2413716479777598E+01
#   1      1      2      8.625216      8.0941033638003375E+01
#-----
#
#.....

```

The variable *it* is a dummy variable for enumerating the temperature values, while, *ik* is the number of 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.

Similiarly, the format for *si.imsigma_mode* is

```

# Electron (Imaginary) Self-Energy in the Migdal Approximatio
n #
# (WARNING: only output a part of the eigenstate
s) #

#=====
#
#NO.k: 450 NO.bands: 2 NO.T: 1 NO.modes: 6
# it ik ibnd E(ibnd)(eV) imode Im(Sigma)(meV)
# Temperature(T)= 25.85203 meV; Chem.Pot.(mu)= 6.55048 eV
  1 1 1 6.955370 1 1.4153504009369593E+00
  1 1 1 6.955370 2 5.7649932505673829E-01
  1 1 1 6.955370 3 3.3560451875664166E+00
  1 1 1 6.955370 4 9.1248572526519600E-01
  1 1 1 6.955370 5 6.7828926275888612E-01
  1 1 1 6.955370 6 5.4750465781934041E+00
  1 1 2 8.625216 1 1.4251118413529920E+01
  1 1 2 8.625216 2 2.3885895867766276E+01
  1 1 2 8.625216 3 1.9149613992834450E+01
  1 1 2 8.625216 4 8.0914088438082139E+00
  1 1 2 8.625216 5 5.5520190253046220E+00
  1 1 2 8.625216 6 1.0010977494759880E+01
#-----
.....

```

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

Remember to converge the Im(Sigma) results with respect to the number of samples (*nsamples*).

(ADD EXPLANATION)

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 $2/\hbar \times \text{ImSigma}$ (it's convenient to use $\hbar = 0.65821195 \text{ eV fs}$ 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 ****.py* (ADD HERE) we provide to post-process the *imsigma* output. 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 “examples/example01-silicon-perturbo/perturbo/pert-imsigma-hole”, 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: examples/example01-silicon-perturbo/perturbo/pert-meanfp-electron (pert-meanfp-hole)

Computes: The e-ph mean free paths for electronic states in a user-defined 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 from the as 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_bin>/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 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)          #
#####
####
#N0.k:      450    N0.bands:      2    N0.T:      1
# it  ik  ibnd  E(ibnd)(eV)  Relaxation time(in f
s)          MFP (in nm)
# Temperature(T)= 25.85203 meV; Chem.Pot.(mu)= 6.55048 eV
1      1      1      6.955370  2.6511488462206518E+01  9.5929
573542019302E+00
1      1      2      8.625216  4.0659982512529345E+00  4.8049
949684520232E+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 3rd, 4th, and 5th 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
  1    2      8.625216   -0.01250  0.58750 -0.01250   -0.043
06 -0.99814 -0.04306   1.1817503775293969E+06
  2    1      6.915451    0.00000  0.60000  0.00000    -0.000
00 -1.00000  0.00000   3.1811354238290834E+05
  2    2      8.510979    0.00000  0.60000  0.00000     0.000
00 -1.00000 -0.00000   1.1197561247449291E+06
  3    1      6.932859   -0.02500  0.60000  0.00000   -0.499
08 -0.86656  0.00000   3.6515768155811669E+05
  3    2      8.520464   -0.02500  0.60000  0.00000   -0.088
88 -0.99604 -0.00000   1.1259209936791812E+06
.....

```

The 1st to 3rd columns are the same as in *prefix.mfp*. The 4th to 6th columns are the 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 “examples/example01-silicon-perturbo/perturbo/pert-meanfp-hole”.

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 approximation (RTA) of the Boltzmann transport equation (BTE) or an iterative approach (ITA) to fully solve the linearized BTE.

Relaxation time approximation (RTA)

Directory: examples/example01-silicon-perturbo/perturbo/pert-trans-RTA-electron (pert-trans-RTA-hole)

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*: set to ‘trans’
- *boltz_nstep*: 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

femper = '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`
- *femper*: 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_bin>/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/
#          #
#-----#

# 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/
#          #
#-----#(for semiconducto
#-----#

# 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 $\sim 1608 \text{ cm}^2\text{V}^{-1}\text{s}^{-1}$, in reasonably good agreement with the experimental value of roughly $1400 \text{ cm}^2\text{V}^{-1}\text{s}^{-1}$.

(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 Fermi Level: 6.550482

6.632612      2.0230556858340154E+01      0.000000E+00      0.000000
0E+00      0.000000E+00      0.000000E+00      0.000000E+00      0.000000
E+00
6.633612      1.9522224579919563E+01      0.000000E+00      0.000000
0E+00      0.000000E+00      0.000000E+00      0.000000E+00      0.000000
E+00
6.634612      1.8836601850198139E+01      0.122626E-03      0.50795
5E-13      0.122626E-03      -0.954877E-13      -0.185696E-12      0.122626
E-03
.....
```

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 is the TDF for each energy, 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 k and q-points, namely the variables *boltz_kdim* and *boltz_qdim*.

An example for hole carriers is also provided, in the folder “examples/example01-silicon-perturbo/perturbo/pert-trans-RTA-hole”.

Iterative approach (ITA)

Directory: examples/example01-silicon-perturbo/perturbo/pert-trans-ITA-electron (pert-trans-ITR-hole)

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

Note: The user needs to run the calculation modes “setup” and then “insigma” since this calculation mode relies on their outputs

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

- `_calc_mode` is set to 'trans'
- `_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'

  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”

```

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

```

It takes ~20 hours using one thread and one MPI process. To speed up the calculations, the user can increase the OpenMP threads and/or the MPI processes. If the number of MPI processes is increased, one has to make sure that the RAM is large enough. After the calculation has completed, we obtain 3 output files, *prefix.cond*, *prefix.tdf*, and *prefix_tdf.h5*, similar to the RTA calculation.

An example calculation for holes is also provided in the folder “examples/example01-silicon-perturbo/perturbo/pert-trans-ITR-hole”.

`calc_mode = 'trans-pp'`

Directory: examples/example01-silicon-perturbo/perturbo/pert-trans-pp-electron

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*
- *femper*: here *si.temper*
- *prefix_tet.h5*: here *si_tet.h5*
- *prefix_tdf.h5*: here *si_tdf.h5*

Run `perturbo.x`:

```
$ mpirun -n 1 <perturbo_bin>/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    0.425885E+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 "examples/example01-silicon-perturbo/perturbo/pert-trans-pp-hole".

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 `perturbo.x` and `qe2pert.x` of this manual.

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

- “pw-ph-wann” for the scf, nscf, phonon, and Wannier90 calculations
- “qe2pert” to generate *prefix_epwan.h5*
- “perturbo” for `perturbo.x` calculations

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”

For each `perturbo.x` calculation, it is essential to always link or copy *prefix_epwan.h5*.

Silicon (with spin-orbit coupling)

📁 **Directory:** examples/example03-silicon-soc

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 complete, 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
qe_band_min = 1
qe_band_max = 32
num_wann = 16
lwannier=.true.
/
```

The input file is similar to the one for silicon without SOC (“examples/example02-silicon-qe2pert”). We only need to double the number of Wannier functions (`num_wann` variable) and DFT bands (`qe_band_min` and `qe_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 *band_min* and *band_max*. Each calculation is the same as in the silicon example without SOC.

GaAs (polar material)

■ **Directory:** examples/example04-gaas-polar

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 similar to the silicon case (“examples/example01-silicon-perturbo”). 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 q-point importance *sampling* and set the variable `cauchy_scale` for the Cauchy distribution. For the short-range (nonpolar) part, we use ‘rmpol’ 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 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 q-points.

Graphene (2D material)

■ **Directory:** examples/example05-graphene-2d

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
  qe_band_min = 1
  qe_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:** examples/example06-aluminum

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”.

Parameters for qe2pert.x

Parameters for

Name	Type	Description
prefix	string	Job name prefix. It should be the same as the prefix used in QE
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
phdir	string	Name of the directory where the phonon "save" directory is located
qe_band_min	intger	Lowest band index used in Wannier90; the default is 1
qe_band_max	intger	Highest band index used in Wannier90; the default is 10000
num_wann	intger	Number of Wannier functions
system_2d	logical	Set it to .true. if the system is 2D; the default is .false.
nk1, nk2, nk3	intger	Number of k points along each dimension used in the Wannierization
debug	logical	Set to .true. to turn on the debug mode (the default is .false.)
lwannier	logical	Set to .true. to rotate the wavefunctions using Wannier unitary matrix before computing e-ph matrix elements (the default is .true.)
load_ephmat	logical	Set to .true. to load prefix_elph.h5 from the directory specified by the variable outdir (the default is .false.)

eig_corr	string	File containing the Kohn-Sham eigenvalues (usually called prefix.eig) prefix.eig
polar_alpha	real	Used only in polar materials. to be consistent with rigid_bulk, enforce alpha=1.0 for lattice dynamical
asr	string	Acoustic sum rule; options: 'no', 'simple', 'crystal' (default)
thickness_2d	real	Thickness of the 2d system, used in the 2D polar e-ph correction. Only needed when system_2d=.true.. The default is 6 (Å)

Parameters for *perturbo.x*

Name	Type	Description
JOB CONTROL		
prefix	string	Job name prefix. It should be the same as the prefix used in QE
calc_mode	string	Calculation mode: 'setup', 'bands', 'phdisp', 'ephmat', 'imsigma', 'meanfp', 'trans', 'trans-pp', 'dynamics-run', 'dynamics-pp' the last two are beta versions
fklist	string	Name of the file containing the k-point list (in crystal unit)
fqlist	string	Name of the file containing the q-point list (in crystal unit)
ftemper	string	Name of the file containing values for the temperature (K), chemical potential (eV), and carrier concentration (cm^{-2} or cm^{-3})
debug	logical	.true. turns on the debug mode; the default is .false.
hole	logical	Set to .true. for calculations on hole carriers; the default is .false.
tmp_dir	string	The directory where the e-ph matrix elements are stored when calc_mode='trans'
load_scatter_eph	logical	Read the e-ph matrix elements from the files in tmp_dir; the default is .false.; used in the calculation mode 'trans'
BOLTZMANN TRANSPORT EQUATION		

boltz_kdim(1:3)	integer	Number of k points along each dimension for the Boltzmann equation; the default is 1.
boltz_qdim(1:3)	integer	Number of q points along each dimension for the Boltzmann equation; the default is boltz_qdim(i)=boltz_kdim(i)
band_min	integer	Lowest band included; the default is 1
band_max	integer	Highest band included; the default is 9999999
boltz_emin	real	Bottom of the energy window (in eV) for the Boltzmann equation; the default is -9999.0
boltz_emax	real	Top of the energy window (in eV) for Boltzmann equation; the default is 9999.0
boltz_nstep	integer	Number of iterations for solving the Boltzmann transport equation
boltz_de	real	Energy step (in meV) for the integrals in the Boltzmann equation; the default is 1.0
delta_smear	real	Smearing (in meV) for the Dirac delta function; the default is 10.0
phfreq_cutoff	real	Phonon energy threshold (meV). Phonons with energy smaller than phfreq_cutoff will be excluded
trans_thr	real	Threshold for the iterative procedure; the default is 0.002
POLAR CORRECTION (required only for calc_mode='imsigma')		

polar_split	string	Three options are: ' ' (leave blank, the default), 'polar', or 'rmpol'; if not specified, the code will compute both the polar and nonpolar parts
sampling	string	Options: 'uniform' (default), 'cauchy'
cauchy_scale	real	Scale parameter gamma for the Cauchy distribution; used when sampling='cauchy'; the default is 0.05
nsamples	integer	Number of q-points for the summation over the q-points in imsigma calculation; the default is 1,000,000
DYNAMICS (via the time-dependent BTE)		
boltz_efield(1:3)	real	External electric field (in V/cm); the default is 0.0
time_step	real	Time step for the carrier dynamics (in fs)
output_nstep	integer	Print out the results every n time steps; the default is 1
boltz_init_dist	string	Initial electron distribution at time zero options: 'restart', 'lorentz', 'fermi', 'gaussian'
boltz_init_e0	real	Energy (in eV) at which the initial distribution is centered. Needs to be specified for boltz_init_dist='lorentz' or 'gaussian'.
boltz_init_smear	real	The broadening or width (in meV) of the initial distribution. Needed for boltz_init_dist='lorentz' or 'gaussian'
solver	string	Options: 'euler' (default), 'rk4'