The SKiES package user manual
Supplemental material for manuscript
"SKiES: the program for *ab initio* calculations of transport properties based on Allen's method for solving Boltzmann equation"

I. S. Galtsov[a,b], V. B. Fokin[a], D. V. Minakov[a], P. R. Levashov[a]
[a]Joint Institute for High Tempaerature RAS
[b]Moscow Institute of Physics and Technology

July 14, 2025

1 Building and Installation

1.1 Quantum Espresso EPW

SKiES is a CMake-built project, so one has first to download and install CMake of minimum version 3.15 from the official website: https://cmake.org/download. one can download SKiES from github by executing:

```
$ git clone git@github.com:JLab-MatSci/SKiES.git
```

The core SKiES folder with source files will be created, let us call it SKiES_ROOT_DIR for the following. Next, SKiES exploits EPW software for different Wannier interpolation routines which is a part of Quantum Espresso (QE) package. The current SKiES version is only compatible with QE of version 7.1 which may be downloaded from https://gitlab.com/QEF/q-e/-/releases/qe-7.1. We recommend one to use the patch UNIX command with additionally provided .patch files located in qe-patches folder of SKiES to disable an annoying standard output and allow for better future use of skies executable. For this enter the root folder of downloaded QE (here and elsewhere let us call it QE_ROOT_DIR) and launch the patch command:

```
$ cd QE_ROOT_DIR/EPW/src
$ patch < SKiES_ROOT_DIR/qe-patches/epw_readin.patch
$ patch < SKiES_ROOT_DIR/qe-patches/io_epw.patch
$ patch < SKiES_ROOT_DIR/qe-patches/wan2bloch.patch
```

To build QE with CMake enter the QE_ROOT_DIR, create the build folder and go into it:

```
$ cd QE_ROOT_DIR
$ mkdir build
$ cd build
```

Now you can choose either a standard CMake command-line interface for the project configuration, or a more convenient GUI tool called ccmake which is also recommended to be installed with the cmake itself. If you choose the latter approach, enter the following command in the build folder:

```
$ ccmake ..
```

You will see the ccmake basic window with the "EMPTY CACHE" greeting line. The basic command here is to enter [c] key to start the configuration process (also press [h] key in case of difficulties with the ccmake interface or see the official docs). The first wave of configuration is launched at this time which automatically sets C, C++ and Fortran compilers and other basic tools in accordance with one's specific PATH variable (one may also activate some preinstalled UNIX module with module load command if such a functionality is provided on a computing cluster). In the end of this step press [e] and the page with many CMake cache variables will emerge. As an example, the following standard variables are set initially by default:

```
CMAKE_BUILD_TYPE                *Release
CMAKE_INSTALL_PREFIX            */usr/local
```

They may be changed by pressing [enter] and manually editing. Let us name the chosen installation folder as QE_INSTALL_DIR. Next, we highly recommend to change the following variable for FFTW library:

```
QE_FFTW_VENDOR                  AUTO
```

to this one:

```
QE_FFTW_VENDOR                  Internal
```

to significantly ease the following step of SKiES installation.

As Wannier routines from wan2bloch.f90 original EPW file do not actially use any MPI functionality, we also recommend build QE with no MPI support to eliminate some possible issues with compilers compatibility at the SKiES building step:

```
QE_ENABLE_MPI                   OFF
```

Actually, when building QE for launching DFT and DFPT part of calculations, for sure choose the version with MPI turned on. So, it is better to have two different versions of QE installed: one for use as third-party libraries in SKiES and the other is for actual *ab initio* calculations. We also advice one to leave all the rest configuration variables initialized with their default values when building for the exclusive use with SKiES.

After configuring the mentioned cache variables press [c] button again until all you see invitation to press [g] key for finally generating the CMake project. After that a Makefile should appear in the build folder. Use make to build and install QE:

```
$ make -j4 install
```

When the process of building reaches 100 %, one should find the QE package fully installed in the QE_INSTALL_DIR folder. Its structure for version 7.1 is the following:

```
QE_INSTALL_DIR
├── bin
├── include
├── share
└── lib
    ├── pkgconfig
    └── cmake
        ├── mbd
        │   ├── MbdConfig-release.cmake
        │   └── MbdConfig.cmake
        └── qe
            ├── qeConfig.cmake
            ├── qeConfigVersion.cmake
            ├── qeTargets-release.cmake
            └── qeTargets.cmake
```

Pay special attention to the files in QE_INSTALL_DIR/lib/cmake/mbd and QE_INSTALL_DIR/lib/cmake/qe folders. These are essential for building SKiES as described below.

1.2 SKiES

After one has successfully installed QE-EPW, building process of SKiES is rather straightforward. The same as above, create a build folder and go into it:

```
$ cd SKiES_ROOT_DIR
$ mkdir build
$ cd build
```

Again, we advise using ccmake routine:

```
$ ccmake ..
```

During the configuration process of SKiES make sure to use the same Fortran compiler as was used in the QE building process. Again, enter the appropriate installation folder (let us call it SKiES_INSTALL_DIR), build type (Release/Debug) and choose either to turn on the following SKiES options:

```
SKIES_ENABLE_DOC                OFF
SKIES_ENABLE_TBB                ON
SKIES_ENABLE_TEST               OFF
```

The first one enables/disables documentation generation for the code structure (files, classes, functions, etc.). For that one has to install Doxygen first. If turned on, some additional stages of configuration are launched to fetch css extensions for more cute html doc pages design (just press [c] until everything is installed, this step requires an Internet access). After choosing ON option for the SKIES_ENABLE_DOC and finishing the generation ccmake stage run the corresponding make target:

```
make doxygen
make -j4 install
```

The documentation is prepared both in html format (open any .html file in SKiES_INSTALL_DIR/docs folder with the help of xdg-open or any browser you like) and in .pdf format as can be found in SKiES_INSTALL_DIR/docs/latex/refman.pdf file.

The second option is turned on by default to support parallel multithreaded version of the code. It is based on C++17 parallel policies for the STL algorithms and requires Intel Threading Building Blocks (TBB) library to be installed beforehand (for details see e.g. https://www.geeksforgeeks.org/execution-policy-of-stl-algorithms-in-modern-cpp/). TBB must also be installed via CMake, the path to the TBBConfig.cmake file must be provided in the following line of the ccmake configuration page:

```
TBB_DIR         TBB_INSTALL_DIR/lib/cmake/tbb
```

The same way one has to enter the paths to the files MbdConfig.cmake and qeConfig.cmake to link with the necessary QE-EPW libs:

```
Mbd_DIR         QE_INSTALL_DIR/lib/cmake/mbd
qe_DIR          QE_INSTALL_DIR/lib/cmake/qe
```

One more SKiES build option is SKIES_ENABLE_TESTS. When turned on, googletest package will be automatically installed (the Internet access is required) and the binary executable called launch_tests will be built. In the end of the full SKiES installation process, one is able to activate this binary to run some basic tests which cover basic SKiES functionalities. It is more important for developers than for ordinary users, but it is a rule of thumb to check that all tests passed immediately after the installation process.

If the configuration CMake step has finished successfully and generation is done, the Makefile for SKiES will emerge in the build folder:

```
$ make -j4 install
```

The structure of SKiES installation folder is the following (the main binary executable is /bin/skies):

```
SKiES_INSTALL_DIR
├── bin
│   ├── skies
│   └── launch_tests
├── include
├── share
│   └── cmake
├── lib
├── docs
│   └── latex
└── examples
```

The /share/cmake folder holds the skiesConfig.cmake file to be used the same way we saw above, namely for other packages to be easily linked to SKiES libraries. So, CMake really provides a convenient way to combine rather different projects into one integrated application and at the same time allows for future extensions.

2 Running examples

In order to demonstrate some features of SKiES functionality, we provide an example of Ag and Pd calculations distributed in separate folders. The folders with the suffix '_not_ready' contain all the files necessary for launching preliminary Quantum ESPRESSO/EPW calculations. See the corresponding README.md files for the details. The other folders (with the "_ready" suffix) hold the files 'crystal.fmt', 'epwdata.fmt' and 'vmedata.fmt' which are the byproducts of the preliminary Quantum ESPRESSO/EPW calculations which are described above. The only file that is missing is the binary heavy file with the suffix '.epmatwp' containing e-ph matrix elements. One has to download it from the remote repository as described in the corresponding README.md files. After this file is downloaded one is able to run the SKiES test calculations in the subfolders 'skies_bands', 'skies_phonons', 'skies_transport_lowT' and 'skies_transport_general' as described in details in the bash scripts given and in the manual.pdf in the root folder of SKiES.

2.1 Quantum Espresso EPW

The input files located in Ag_no_epw_prep are prepared for launching QE-EPW *ab initio* calculations. This example corresponds to the one described in our paper and one can all the set of parameters we used to obtain the final results for transport properties. First, one should launch the relax.sh script (an example of sbatch script is given which is to be configured for one's specific system, at least path to one's parallel QE binaries) to obtain the lattice parameter of a relaxed fcc Ag structure (see the output relax.out file for the new volume of the unit-cell volume).

Next, see the file run_qe.sh as an example of sequential steps for obtaining data on electron-phonon interaction in the Wannier representation. We do not stop here for a thorough description of each step. Much more details may be found in the tutorial carefully prepared by the EPW developers group for several previous EPW Summer Schools (see e.g. https://docs.epw-code.org/doc/School2022.html, exercise 2 of Wednesday Hands-On tutorial 1 prepared by S. Poncé). We strongly follow the steps given in this tutorial. Do not forget to change the prefix ('ag' in this example) in all the .in files and in pp.py when changing the material of interest. The only noticeable difference is a much less verbose epw.in file. The minimal example of this input file is given below:

```
--                                                                              epw.in
&inputepw
  prefix      = 'ag'
  amass(1)    = 107.87
  outdir      = './'
  dvscf_dir   = './save'

  elph        = .true.
  epbwrite    = .true.
  epbread     = .false.
  epwwrite    = .true.
  epwread     = .false.
  vme         = 'wannier'

  nbndsub     =  9
  bands_skipped = 'exclude_bands = 10-11'

  wannierize  = .true.
  num_iter    = 500
  dis_win_max = 80.0
  dis_froz_max = 50.0

  nk1         = 6
  nk2         = 6
  nk3         = 6

  nq1         = 6
  nq2         = 6
  nq3         = 6
 /
```

The special attention should be paid to the values of elph, epbwrite, epbread, epwwrite and epwread tags.

Their values must be changed appropriately later when launching SKiES calculations. The same must be said about the Wannier interpolation tag wannierize. Also remember the nbndsub value to be in agreement with the value of bands_skipped and with the number of bands nbnd given in nscf.in file. Finally, one has only to provide values of nk1, nk2, nk3 for the coarse grid of **k**-points (the same as in nscf.in file) and nq1, nq2, nq3 for the coarse grid of **q**-points (the same as in ph.in file). There is no need to provide nkf1, nkf2, nkf3 and nqf1, nqf2, nqf3 or any other EPW parameters for self-energies and other kinds of calculations because the only mission of EPW in the context of SKiES calculations is to generate output files prefix.epmatwp, vmedata.fmt, crystal.fmt and epwdata.fmt. These files contain all the necessary information about electron, phonon properties and electron-phonon interaction in the Wannier representation. Note that the EPW part of the overall QE calculations will interrupt with the following error:

```
                                                                                                    epw.out
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    Error in routine loadqmesh_serial (1):
    Cannot load fine q points
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    stopping ...
```

which is ok, because the mentioned files are written to the disk at this step. More details about the input parameters of EPW package and its interaction with the wannier90 package may be found on the official EPW page: https://docs.epw-code.org/index.html.

2.2 SKiES

Most files and subdirectories located in Ag_epw_prep folder are byproducts of QE-EPW calculations conducted as described in the previous subsection. There are really many of them as EPW calculations rely on numerous calls to other programs. Fortunately, one does not need to know anything about structure of the folder when launching SKiES routines inside of it. The only requirement is the presence of files generated by EPW itself as described in the previous section.

First of all, we recommend create a new folder where all the SKiES output files will be stored not to be mixed with the QE-EPW files. For example, enter the Ag_epw_prep folder and create a subdirectory named skies:

```
$ mkdir skies
$ cd skies
```

Then one has to copy the epw.in file described above into this subdirectory and rename it as epw.skies.in file. Now change several lines to obtain the following one:

```
--                                                                                                  epw.skies.in
&inputepw
  prefix      = 'ag'
  amass(1)    = 107.87
  outdir      = '../'
  dvscf_dir   = '../save'

  elph        = .true.
  epbwrite    = .false.
  epbread     = .false.
  epwwrite    = .false.
  epwread     = .true.
  vme         = 'wannier'

  nbndsub     =  9
  bands_skipped = 'exclude_bands = 10-11'

  wannierize  = .false.
  num_iter    = 500
  dis_win_max = 80.0
  dis_froz_max = 50.0
```

```
  nk1         = 6
  nk2         = 6
  nk3         = 6

  nq1         = 6
  nq2         = 6
  nq3         = 6
 /
```

At this step the `wannierize` flag is turned off, `epwread` tag if set to `.true.`, `epwwrite` is `.false.` and `epbwrite` is `.false.`. Now one is able to launch `SKiES` commands described in details in the next section and in the manuscript itself. Also note the changes made to the `outdir` and `dvscf_dir` lines: here one has to provide either a relative (as in this example) or a full path to the files `prefix.epmatwp`, `vmedata.fmt`, `crystal.fmt` and `epwdata.fmt` generated by `EPW`.

3 SKiES commands

To see the list of all available commands, execute the following command:

```
$ skies list
```

The help query may be launched for each of the commands, e.g. for the `dos` command one should execute

```
$ skies help dos
```

The main `SKiES` commands are enumerated below with the minimal (mandatory) list of input options. The full list of options may be seen with `skies help <cmd-name>` command.

The `dos` command evaluates electronic density of states (DOS):

```
$ skies dos --grid=[n1,n2,n3] < epw.skies.in
```

where $n1, n2, n3$ are three integer numbers denoting the dimensions of desirable Monkhorst-Pack grid in the 1st BZ. The `phdos` (phonon DOS) and `trdos` (transport DOS) commands have the similar view:

```
$ skies phdos --grid=[n1,n2,n3] < epw.skies.in
$ skies trdos --grid=[n1,n2,n3] < epw.skies.in
```

An important option which we highly recommend to use is `--tetra`. It uses The calls to the mentioned commands internally exploit the tetrahedron method for BZ sampling, as described in [1], [2]. By default another approach is chosen, namely the gaussian smearing for Dirac delta-functions with $\sigma_{el} = 0.03$ eV and $\sigma_{ph} = 0.5$ meV. More details may be found in the manuscript, here we only would like to mention that the latter approach requires convergence both over the **k**-point grid density and the values of σ_{el} and σ_{ph} while the tetrahedron method does not depend on smearing parameters at all and usually provides much more detailed results for specific areas of the 1st BZ.

The next group of commands allows one to calculate dispersion relations along a high symmetry path in the reciprocal space. For example, `bands` command evaluates a band electronic structure:

```
$ skies bands < epw.skies.in
```

This command requires a `KPATH` file being defined in the calculation folder. This file has the following format (an example is given for a FCC lattice):

```
                                                                    KPATH
G 0.000, 0.000, 0.000
X 0.000, 0.500, 0.500
W 0.250, 0.750, 0.500
K 0.375, 0.750, 0.375
G 0.000, 0.000, 0.000
```

First column contains user-defined labels for **k**-points for which three crystal coordinates (i.e. in the basis of $\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3$ reciprocal lattice vectors) follow separated by a space and a comma. After the calculation is finished, one obtains a file called `KLABELS` in the following form:

```
                                                                    KLABELS
# Label    Distance [Angstrom]
      G                      0
      X                1.54695
      W                2.32043
      K                2.86736
      G                4.50815
```

which contains the distance along the chosen k-path in Å. Another output of the `bands` command is the file named `EigenValue.dat` which contains the desired Wannier-interpolated electronic band structure. The number of evaluated bands corresponds to the number of Wannier orbitals chosen in `epw.in` file.

The same description is valid for the `phonons` and `velocs` commands. The only command with a bit different user interface is `elphmat`: to obtain matrix elements of electron-phonon interaction one has to additionally provide the following parameters: the initial band number (starting from zero) `--band-ini`, the final band number `--band-fin` and the desired phonon dispersion branch `--phon-branch`. Also one can customize the initial **k**-point via `--kpoint-ini`. For example, the following command will launch calculations of $g_{nm\nu}(\mathbf{k}, \mathbf{q})$ with $\mathbf{k} = \mathbf{0}$ (Γ-point, set by default), $n = 3$ (initial band), $m = 0$ (final band) and $\nu = 1$ (phonon branch):

```
$ skies elphmat --band-ini=3 --band-fin=0 --phon-branch=1 < epw.skies.in
```

The central command of `SKiES` is `a2f`. It is used to evaluate transport spectral function $\alpha^2_{tr}F$ (see the manuscript for details) and supports two independent approaches. The first approach is a low temperature approximation ($\alpha^2_{tr}F(s, s', \alpha, \beta, \Omega)$ only depends on a range of phonon frequencies Ω) and the second one is the general Allen's method [3] ($\alpha^2_{tr}F(s, s', \alpha, \beta, \varepsilon, \Omega)$ depends both on electronic and phonon energies). While in the first approach many simplifications are made which significantly accelerate the calculations, its reliability may be quite low in cases of complicated Fermi surfaces. In this approach only one single value of electronic energy participate in the calculation, namely the Fermi level which is a mandatory input option `--eF`. Also the minimal command requires both signs s, s' to be provided and the grids of fine **k**- and **q**-points to use in the Wannier interpolation. For example, to execute a calculation for $\alpha^2_{tr}F(+1, +1, z, z, \Omega)$ in the range of phonon energies from 0.1 meV to 25 meV, enter the following command:

```
$ skies a2f --signs=[1,1] --eF=12.9148 --kgrid=[k1,k2,k3] --qgrid=[q1,q2,q3] \
>           --alpha=z --beta=z --omegas=[0.1,25] < epw.skies.in
```

where `ki` and `qi` stand for the dimansions of the desired MP grids and the value of Fermi energy which should be taken from a scf `Qunatum Espresso` calculation, e.g. using the following command:

```
$ grep Fermi scf.out
the Fermi energy is    12.9148 ev
```

As in the group of DOS commands, we highly recommend one to always use the `--tetra` option for turning the tetrahedron method on when sampling the 1st BZ. The alternative are the couple of `--sampling` and `--smearing` options.

An important note should also be made for the `--bands` option. We highly recommend to always draw a band structure before launching actual `a2f` calculations. As shown in the manuscript (fig. 2), only one wide band is crossed by the Fermi energy, so it is very ineffective to consider any other bands in the spectral function evaluation. As the numeration of bands starts from zero, the 5th band is crossed in this example for Ag, and one has to additionally write:

```
$ skies a2f --signs=[1,1] --eF=12.9148 --kgrid=[k1,k2,k3] --qgrid=[q1,q2,q3] \
>           --bands=[5,5] < epw.skies.in
```

The second approach requires a list of electronic energies in the format `--epsilons=[low,high]`, where low stands for the lower bound relative to the Fermi level and high is for the upper bound. For example, if one wants to evaluate transport spectral function $\alpha^2_{tr}F(-1, +1, x, x, \varepsilon, \Omega)$ for the range of electronic energies from $\varepsilon - 0.5$ eV to $\varepsilon_F + 0.5$ eV, the following command helps:

```
$ skies a2f --signs=[-1,1] --eF=<eF> --kgrid=[k1,k2,k3] --qgrid=[q1,q2,q3] \
>           --epsilons=[-0.5,0.5] < epw.skies.in
```

After launching any of the `a2f` command versions mentioned above, a file with the name `LambdaTr_`$ss'\_\alpha\beta$`.dat` is opened for writing. The name contains one of the letters p, m in positions of s, s' encoding the first and the second (order is essential!) signs of $\alpha^2_{tr}F(s, s', \alpha, \beta, ...)$. As one might guess, p denotes $+1$ and m does for -1. After the second underscore two letters from the set of x, y or z follow which stand for the chosen cartesian components. This file structure is described below. The main purpose of it is logging and the opportunity to continue an interrupted calculation. Also note that in the case of general Allen's method files `FSH_`x`.dat`, `FSH_`y`.dat` and `FSH_`z`.dat` are written to disk. They contain Fermi surface harmonics used in the calculations (see the manuscript for details) and are required to continue a calculation. If some calculation was interrupted, just enter the following command in the same folder:

```
$ skies a2f --signs=[1,-1] --continue < epw.skies.in
```

to continue that calculation. Note one has only to explicitly specify the signs of the certain spectral function which calculation was interrupted as there might be several calculations in parallel running in the same folder (with the same epw.skies.in file). All the other parameters are automatically read from the interrupted `LambdaTr` file.

After the calculation is successfully finished, a file `SpecFunc_`$ss'\_\alpha\beta$`.dat` is written to the disk. The foramt is the same as for the corresponding `LambdaTr` file. It contains results for transport spectral function, the details of its format see below.

Note that every transport coefficient calculation supported in `SKiES` requires a different set of `SpecFunc` files being calculated first. The table 1 is made to clarify each case (also see the manuscript).

Table 1. The Requirements for Transport Properties Calculations

Transport coefficient	Minimal set of SpecFunc files	Odd corrections
$\rho_{\alpha\alpha}$ (Resist_$\alpha\alpha$.dat)	SpecFunc_pp_$\alpha\alpha$.dat	no
$\kappa_{\alpha\alpha}$ (ThermalCond_$\alpha\alpha$.dat)	SpecFunc_pp_$\alpha\alpha$.dat, SpecFunc_mm_$\alpha\alpha$.dat	SpecFunc_pm_$\alpha\alpha$.dat, SpecFunc_mp_$\alpha\alpha$.dat

After all the necessary transport spectral functions are evaluated as described in the table 1, one is able to continue with the final transport properties calculations. There are three main commands: `resist` (calculates electrical resistivity) and `thermal-cond` (thermal conductivity). An example for a minimal `resist` command is given below:

```
$ skies resist --range=[10,1500] < epw.skies.in
```

It only takes one mandatory option for temperature range specification in the one-temperature regime (i.e. when $T_e = T_i$ in formulas for transport coefficient the manuscript). If one is also interested to include a phononic temperature T_i, an additional range `--iont-range` must be provided. By default the command evaluates the xx-component of resistivity, but the option `--alpha` may be explicitly given from a set of x, y or z. Remember the differences in calculations of transport coefficients using the low temperature approximation (when only one electronic energy value is used at the Fermi level) and using the general Allen's method formulas (where there is a finite range of electronic energies ε explicitly given). By default the above command assumes the low temperature case and requires the corresponding `SpecFunc` files only contain values for the Fermi energy level (equals to zero by agreement). If the calculations were launched for general Allen's transport spectral functions, please do not forget to explicitly provide the `resist` command with the key `--general`.

The command line interface for the `thermal-cond` command is the same, but please check the table 1 not to forget all the kinds of $\alpha^2_{tr}F$ files which must be generated first by the `a2f` command.

4 EigenValue.dat, EigenFrequency.dat, Velocities.dat and EPHMatrix.dat files

An example of an `EigenValue.dat` file is given here. The other files look mostly the same.

```
                                                                EigenValue.dat
 # Kpath [1 / Angstrom]       EigenValue dispersion [eV]
```

```
              0.05016      -3.223       20.44       20.46       20.48
               0.1003       -3.19       20.25       20.29       20.33
               0.1505      -3.135       19.96       20.02       20.08
               0.2006       -3.06       19.59       19.68       19.75
               0.2508      -2.965       19.17       19.27       19.35
               0.3009      -2.854       18.73       18.83        18.9
          ...
```

The first column contains the k-point distances along the specified path of **k**-points. The other `nbndsub` (see `epw.in` file) columns are filled with the Wannier-interpolated electronic energies (eigenvalues of Kohn-Sham Hamiltonian).

5 EigenValueDOS.dat, EigenFrequencyDOS.dat, VelocitiesDOS.dat files

An example of an `EigenValue.dat` file is given here. The other files look mostly the same.

EigenValueDOS.dat

```
# Energy, eV          EigenValue DOS [1 / eV]
         -10                        0.0142375
    -9.93355                        0.0190848
    -9.86711                        0.0246639
    -9.80066                        0.0257638
    -9.73422                        0.0272332
...
```

The first column contains a specified energy list and the second column contains the corresponding DOS values for these renergies.

6 LambdaTr and SpecFunc files

An example of `SpecFunc_pp_xx.dat` file for low temperature case is given below:

LambdaTr_pp_xx.dat

```
# Mode-resolved transport coupling strength
# num. of q-points: 16x16x16, num. of k-points: 32x32x32
# num. of modes: 3
# low_band: 5
# high_band: 5
# Fermi level: 12.9148 eV
# electron sampling: tetrahedra
# phonon sampling: tetrahedra
# sign: 1
# sign': 1
# velocity component alpha: x
# velocity component  beta: x
# electron energy list [eV]: 0
# electronic DOS list [1/eV/spin/cell]: 0.132128
# transport DOS list [Ry^2*bohr^2/eV]: 0.0754247
# iq       qx      qy      qz    nu        om_q_nu    lambda_q_nu      inner_sum
1      -0.469  -0.469  -0.469     1        0.00979       0.012948        0.31344
1      -0.469  -0.469  -0.469     2        0.00979       0.036058        0.87287
1      -0.469  -0.469  -0.469     3         0.0221       0.028964         1.5849
2      -0.469  -0.406  -0.469     1        0.00999        0.01189        0.29351
2      -0.469  -0.406  -0.469     2         0.0102       0.050239         1.2656
2      -0.469  -0.406  -0.469     3         0.0218       0.029097         1.5709
3      -0.469  -0.344  -0.469     1         0.0105       0.014433        0.37501
3      -0.469  -0.344  -0.469     2         0.0114       0.070121         1.9734
```

```
3      -0.469  -0.344  -0.469     3         0.0213       0.040775         2.1451
...
```

The header contains the main information about the launched calculation, including densities of **k**- and **q**-point grids, number of phonon modes, the given value of the Fermi level etc. Note that the electron energy list only contains one value equal to zero as all the electron energies are evaluated relative to the Fermi level, which we use as the zero reference point. The values of electronic and transport DOS are also given only at the Fermi energy.

The main part of the file contains several columns and each line in this table corresponds to a specific combination of a **q**-point index (`iq`) and a phonon branch ν (`nu`). Also the crystall coordinates of each **q**-point are given. The 6th column contains values of phonon frequencies for each $\mathbf{q}\nu : \omega_{\mathbf{q}\nu}$. The following columns are split into pairs in the case of the general Allen's formulas, the number of such pairs equals to the number of electron energies supplied via `--epsilons` option.

Next, one can see an example of a `SpecFunc_pp_xx.dat` file obtained for a low temperature case. The header is rather self-explanatory. Then two columns follow: the first with the phonon frequencies and the second with the values of the transport spectral function.

SpecFunc_pp_xx.dat

```
# elec_smearing: tetrahedra
# phon_smearing: tetrahedra
# sign: 1
# sign_pr: 1
# velocity component alpha: x
# velocity component beta: x
# electron energy list [eV]: 0.000000
# DOS for energy list [1/eV/spin/cell]: 0.132128
# transport DOS for energy list [Ry^2*bohr^2/eV]: 0.075425
#
# Frequency [eV]        Transport Spectral Function
0.0001                           0
0.000348259                      0
0.000596517                      0
0.000844776                      0
0.00109303              4.27271e-06
0.00134129               2.3099e-05
0.00158955              5.84548e-05
0.00183781              0.000111992
...
```

The same file format is used for spectral functions obtained in the general Allen's method. The only difference is that there are several values of electron energies, DOSes and the corresponding number of columns with spectral functions values.

7 Resist.dat, ThermalCond.dat files

These files names are formed according to the rule similar to the one described above for `LambdaTr` and `SpecFunc` files. After the underscore two cartesian indices follow which correspond to the desired component of resistivity tensor (e.g. `Resist_xy.dat` or `ThermalCond_zz.dat`). An example for `Resist_xx.dat` file is given below (the other types of files look mostly the same):

Resist_xx.dat

```
#  elec_smearing: tetrahedra
#  phon_smearing: tetrahedra
#  Transport DOS per spin [r.a.u.] in energy list: 1.77561 1.75433 1.73744 1.72301 ...
#
# Te [K]      Resistivity [muOhm cm]
10            7.89734e-05
14.9254       0.000710252
19.8507       0.00253625
24.7761       0.00615901
```

```
29.7015       0.0122192
34.6269       0.0213922
39.5522       0.0343619
...
```

First two lines contain information about the type of BZ sampling chosen for this particular calculation. In the case of Dirac delta-functions smearing smearing parameters are given in eV. Then a long line with follows with transport densities of states evaluated for each electronic energy provided in the corresponding spectral functions calculations (note that this list only conatins one element if low temperature approximation was used). Next one obtains two columns for resistivity in the specified temperature range. If one is also interested in two-temperature regime (for different lists of electronic and ionic (phonon) temperatures as may be specified by `--iont-range` option), the file takes the following form:

```
                                                                                Resist_xx.dat
#  elec_smearing: tetrahedra
#  phon_smearing: tetrahedra
#  Transport DOS per spin [r.a.u.] in energy list: 1.77561 1.75433 1.73744 1.72301 ...
#
# Resistivity [muOhm cm]
#  Te [K]  \  Ti [K]:        10        14.9254    19.8507       24.7761   ...
          10               7.89734e-05 0.000573249  0.00259417   0.0078386   ...
     14.9254               0.000379088 0.000710252  0.00206427   0.0055782    ...
     19.8507                0.00126918  0.00151818  0.00253625  0.00517848    ...
     24.7761                 0.0030265    0.003226   0.0040417  0.00615901   ...
     29.7015                0.00594818   0.0061146  0.00679511  0.00856195   ...
     34.6269                 0.0103502    0.010493   0.0110769   0.0125935   ...
...
```

Each element in the obtained matrix serves for the corresponding value of $\rho_{xx}(T_e, T_i)$.

There is also a specific option to be included in the case of general Allen's formulas called `--add_odd` (see the manuscript for details). This flag turns on corrections to thermal conductivity but requires some additional precalculated files with transport spectral functions with $s \neq s'$ (see the table 1). There is no such corrections for electrical resistivity.

References

[1] Ph Lambin and Jean-Pol Vigneron. Computation of crystal green's functions in the complex-energy plane with the use of the analytical tetrahedron method. *Physical Review B*, 29(6):3430, 1984.

[2] PB Allen. A tetrahedron method for doubly constrained brillouin zone integrals application to silicon optic phonon decay. *physica status solidi (b)*, 120(2):529–538, 1983.

[3] PB Allen. New method for solving boltzmann's equation for electrons in metals. *Physical Review B*, 17(10):3725, 1978.