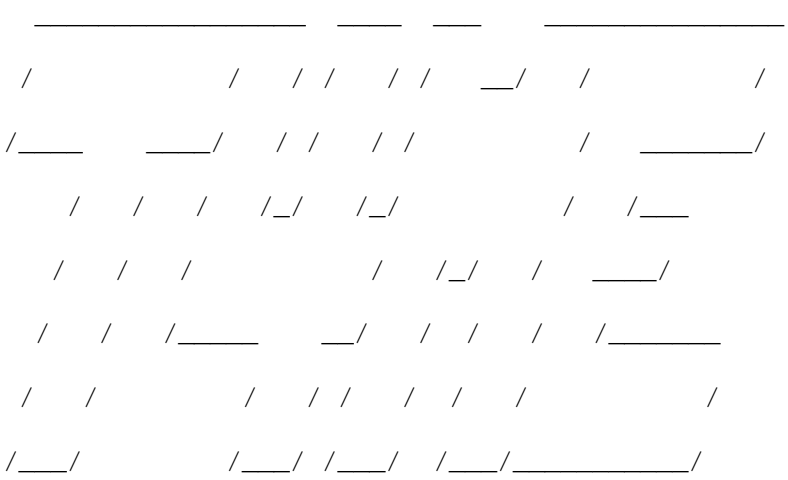
T4ME Documentation

Release 1.1.1

Espen Flage-Larsen

Nov 21, 2019

Contents



Routines to calculate the transport properties of materials using the linearized Boltzmann Transport Equations (BTE) in the Relaxtion-Time-Approximation (RTA).

Please go to the T4ME documentation for more extensive documentation and information regarding usage (the API documentation is currently not operational).

CHAPTER 1

Features

- Modular, easily extendable by users
- Band structures:
  - Generate the band structure from analytic function
    - Parabolic bands
    - Parabolic bands pluss a quartic correction
    - Kane type of bands
  - Read from first-principle codes
    - Interface to VASP is included
    - Interface to read Wannier90 input and output files and use these to construct tight binding orbitals using PythTB is included.
  - Read from NumPy datafiles
- Scattering properties:
  - Parabolic energy dispersion models:
    - Acoustic phonon scattering from deformations
    - Non-polar optical phonon scattering (not fully tested)
    - Piezoelectric acoustic phonon scattering (not fully tested)
    - Polar optical phonon scattering (not fully tested)
    - Intervalley phonon scattering (not fully tested)
    - Ionized impurity scattering
  - Density of states models: - Acoustic phonon scattering from deformations - Non-polar optical phonon scattering (not fully tested) - Polar optical phonon scattering (not fully tested) - Intervalley phonon scattering (not fully tested)
  - Alloy scattering

- Solution of the transport and density of states integrals:
  - Trapezoidal, Simpson and Romberg integration of a static input grid
  - Linear tetrahedron method (Spglib needed)
  - Weighed sum method
- Interpolation of the band structure and scattering properties:
  - All routines available in SciPy
  - GeometricTools/WildMagic regular grid routines

CHAPTER 2

Structure

The structure of the program is simple: the main routines are written in Python utlizing NumPy and SciPy where necessary. In addition there are calls to external routines through Cython, particularly the optional libraries. Only support for Python3 is confirmed.

CHAPTER 3

Contributing and versioning

Standard Git versioning is utilized. Contributions are welcome, encouraged and (greatly) appreciated. Please go here: T4ME@GitHub

CHAPTER 4

Author

Espen Flage-Larsen with funding from the Norwegian Research Council, Thelma project (228854).

CHAPTER 5

License

This project is licensed under the BSD 3-clause license. Please see `LICENSE.md` included in the root folder of T4ME for additional details.

CHAPTER 6

Documentation

6.1 Prerequisites

In its basic form T4ME only need the following dependency:

- Spglib. This can be installed with

```
pip install spglib
```

Additional optional dependencies include:

- Spglib, A custom interface to Spglib to enable tetrahedron integration.
- GeometricTools (use Wildmagic 5.14). Used to interpolate band structure data, for instance by offering Akima interpolation.

6.2 Download

If using *pip* it is not necessary to download the package, it can simply be installed using

```
pip install T4ME
```

Otherwise, T4ME is hosted at GitHub and can be obtain by

```
git clone git@github.com:espenfl/t4me.git
```

or

```
git clone https://github.com/espenfl/t4me.git
```

6.3 Installing

6.3.1 Basic install

First make sure Spglib is installed

```
pip install spglib
```

Then install T4ME by executing the command

```
pip install T4ME
```

This will give the user the posibility to calculate the transport coefficients using integration routines in SciPy. For other integration and interpolation routines the user needs to follow the following recipe.

6.3.2 Advanced install

For more advanced functionality (interpolation and other integration routines) the user should determine which external libraries are needed and install them based on their respective documentation. Please also fetch the repository from github and work from its base directory when executing the following commands.

The `setup.py` file assume in its supplied form that the user installs the libraries in the standard folders, e.g. *$HOME/include* and *$HOME/lib* for the include and library files, respectively. If other locations are needed, please adapt the `setup.py` file.

As an example, we want to enable the tetrahedron integration. A Spglib interface needs to be compiled. This can be build with the included `build_spglib` file.

```
./build_spglib
```

If that was successfull, T4ME can then be built by issuing the following command

```
pip install .
```

or

```
pip install -e .[dev]
```

Another example. We want to enable SKW interpolation. The SKW routines can be built (assuming Intel MKL is installed) by issuing

```
./build_skw
```

If successfull, T4ME can then be installed by issuing one of the two commands listed above. If other FFT routines are to be used, please modify `skw/Makefile`.

All other libraries need to be built externally and linked in.

Upon successfull completion of the installation, T4ME is executed with the command

```
t4me
```

An *input* directory is needed which should contain the input files.

6.4 Running tests

Several tests are included, which can be used to test the installation.

Currently only basic functionality is tested and developers are encouraged to write tests for any added functionality.

Tests are executed by issuing

```
pytest
```

In the base directory.

6.5 Input

The program relies on three parameter files written in YAML. They should all reside inside the *input* directory.

- `input/param.yml` - contains the main parameters, which routines to execute, what type of integration, interpolation etc. to perform and what input files to use.
- `input/bandparam.yml` - contains parameters for band generation, the scattering parameters for each band and the parameters to use in the tight binding generation (if that is needed).
- `input/cellparam.yml` - contains details of the unit cell, its atoms and the reciprocal sampling density.

In addition if external input files be used they should also be placed in the *input* folder.

6.5.1 The input parameters

General input parameters

```
      - carrier_valence_energy
      - carrier_conduction_energy
      - carrier_dos_analytick
      - defect_ionization
      - donor_number
      - donor_degen_fact
      - donor_energy
      - acceptor_number
      - acceptor_degen_fact
      - acceptor_energy
      - read
      - readfile
      - scissor
      - symprec
      - libinfo
      - onlytotalrate
      - parallel
      - run_tests
```

Notes about format

The input files follow normal YAML conventions. Please inspect the sample file `input/param.yml`. Even though many parameters have default values if not specified the user should always run the calculations with fully specified input files for consistency and reproducibility.

Dispersion relations

The following parameters are related to the energy and velocity dispersion relations.

`dispersion_interpolate`

If set to `True` the band structure is interpolated on a k-point grid.

Example:

```
dispersion_interpolate: False
```

Do not interpolated the band structure.

dispersion_interpolate_sampling

The target k-point sampling when performing interpolation.

Example:

```
dispersion_interpolate_sampling: [45,45,45]
```

Interpolates the input band structure to a grid density of 45, 45 and 45 k-points along the unit axis of the supplied k-point grid.

dispersion_interpolate_step_size

The target k-point step size in inverse AA. In order for this parameter to work, the user have to set

```
dispersion_interpolate_sampling: [0,0,0]
```

Example:

```
dispersion_interpolate_sampling: [0.1,0.1,0.1]
```

Creates a k-point sampling that is at least as dense as to give a step size of 0.1 inverse AA between each k-point along each reciprocal axis.

dispersion_interpolate_method

Choses which interpolative method to use. The following options are currently available:

- *linearnd* - Uses `LinearNDInterpolator` in SciPy.
- *interpn* - Uses `interpn` in Scipy.
- *rbf* - Uses the Scipy version, but that is memory intensive
- *wildmagic* - Uses the GeometricTools (former WildMagic) interpolation routines.
- *skw* - Uses Fourier interpolation
- *tb* - Extracts the energies on a denser grid from a tight- binding model

Tests have shown that the last three methods are quite general and, given what they are, quite accurate.

Example:

```
dispersion_interpolate_method: "wildmagic"
```

Will for instance use the WildMagic library.

dispersion_interpolate_type

Additional selective layer for the method chosen by :ref'*dispersion_interpolate_method*. Currently, the following options are availble:

- *nearest* or *linear* - if `dispersion_interpolate_method` = *linearnd*
- *trilinear*, *tricubic_exact*, *tricubic_bspline*, *akima* - if `dispersion_interpolate_method` = *wildmagic*

Example:

```
dispersion_interpolate_type: "akima"
```

Uses the Akima interpolation in the WildMagic library.

dispersion_velocities_numdiff

Use numerical differentiation to calculate the velocities if they are not present on entry, or/and use numerical differentiation to extract the velocities after the dispersions have been interpolated (used by default for the interpolat routines that do not support velocity extraction)

Example:

```
dispersion_velocities_numdiff: False
```

Turns for instance of the numerical difference calculation of the velocities. In this case please make sure that the velocities are present on input or that they are genrated by other means.

dispersion_write_preinter

Selects if a line extraction of the band structure is written to the file `bands` before interpolation. If velocities are present this is also written to the file `velocities`

Example:

```
dispersion_write_preinter: False
```

Writes the extracted band structure values along a line to file(s).

dispersion_write_postinter

Selects if a line extraction of the band structure is written to the file `bands_inter` after interpolation. If velocities are present this is also written to the file `velocities_inter`

Example:

```
dispersion_write_postinter: False
```

Does not write the extracted band structure values along a line to file(s).

dispersion_write_start

The start point (in direct coordinates) for the line extraction.

Example:

```
dispersion_write_start: [0.0, 0.0, 0.0]
```

An example start point, here the Gamma point.

dispersion_write_end

The end point (in direct coordinates) for the line extraction.

Example:

```
dispersion_write_end: [0.5, 0.0, 0.0]
```

dispersion_num_kpoints_along_line

How many samples to use along the line to be extracted.

Example:

```
dispersion_num_kpoints_along_line: 20
```

Here 20 points is used along the line.

dispersion_effmass

Calculate the effective mass tensor along the unit vectors of the configured reciprocal cell. The resulting tensor is in units of the free electron mass. Currently it is not printed out and an error will occur.

Example:

```
dispersion_effmass: False
```

Do not calculate the effective mass tensor.

dispersion_effmass_diagonalize

Diagonalize the calculated effective mass tensor. Currently the diagonal elements and the eigenvectors are not printed out and an error will occur.

Example:

```
dispersion_effmass_diagonalize: False
```

Do not diagonalize the effective mass tensor.

dispersion_effmass_transform

The transformation vectors for the effective mass tensor. The elements [0,:] give the first vector, [1,:] the second and [2,:] the third. Should be given in direct coordinates. If the array is left empty, no transformation is performed.

Example:

```
dispersion_effmass_transform: []
```

Do not transform the effective mass tensor.

Electron transport

The following parameters determines how the transport of electrons is to be determined.

transport_calc

Determines if the transport calculations are to executed.

Example:

```
transport_calc: True
```

Calculate the transport properties.

transport_method

Selects which mode to use to calculate the transport properties. Currently three different modes are accepted;

- *closed* - The integrals are solved using the closed Fermi-Dirac integrals. Only available if the band structure is generated by means of analytic models. Only one scattering mechnism can be used for each band in this approach.
- *numeric* - A numerical integration of the Fermi-Dirac integrals, which allows to concatenate different scattering mechanisms for each band.
- *numerick* - The integrals are solved by integrating over the k-point grid or by utilizing the spectral function.

Example:

```
transport_method: "numerick"
```

In this example the transport integrals are solved using the closed analytical expressions for the Fermi-Dirac integrals.

transport_integration_method

Selects which method to use for solving the integral over the k-points. Only applicable if `transport_method` is set to *numerick*.

- *trapz* - Use the trapezoidal integration scheme implemented in SciPy
- *simps* - Use the Simpson integration scheme implemented in SciPy
- *romberg* - Use the Romberg integration scheme implemented in SciPy
- *tetra* - Use the linear tetrahedron method
- *smeared* - Use the weighted sum approach with a smearing factor

transport_integration_spectral_smearing

Gaussian smearing factor for the weighted sum approach. In units of eV. Only relevant if `transport_integration_method` is set to *smeared*.

Example:

```
transport_integration_spectral_smearing: 0.1
```

Would set it to 0.1 eV.

transport_integration_spectral_density

The sampling density of the spectral function. Only relevant if `transport_integration_method` is set to *tetra* or *smeared*.

Example:

```
transport_integration_spectral_density: 1000
```

An example requesting 1000 samples.

transport_integration_spectral_energy_cutoff

Determines the extra padding that is used for the spectral function on both sides of the requested chemical potential. If multiple chemical potentials are requested, the lowest and the highest value is checked and the range of the energy interval on which the spectral function is calculated is padded with the specified value. Only relevant if `transport_integration_method` is set to *tetra* or *smeared*. In units of eV.

Example:

```
transport_integration_spectral_energy_cutoff: 1.0
```

Here, 1.0 eV is subtracted (added) to the smallest (largest) requested chemical potential.

transport_chempot_min

The minimum chemical potential requested for which the transport coefficients are calculated. In units of eV.

Example:

```
transport_chempot_min: -1.0
```

Starts the calculation of the transport properties at -1.0 eV.

transport_chempot_max

The maximum chemical potential requested for which the transport coefficients are calculated. In units of eV.

Example:

```
transport_chempot_max: 1.0
```

Ends the calculation of the transport properties at 1.0 eV.

transport_chempot_samples

The number of chemical potential samples to use between `transport_chempot_min` and `transport_chempot_max`.

Example:

```
transport_chempot_samples: 100
```

Extract the transport coefficients at 100 points between `transport_chempot_min` and `transport_chempot_max`.

transport_energycutband

Bands that reside `transport_energycutband` outside the chemical potential is dropped from the calculation of the transport coefficients. All k-points are currently analyzed in order to determine which bands fall inside the energy range [`transport_chempot_min`-`transport_energycutband`,‘‘transport_chempot_max‘‘+‘‘transport_energycutband‘‘] . Units in eV.

Example:

```
transport_energycutband: 1.0
```

Substract and add 1.0 eV to `transport_chempot_min` and `transport_chempot_max`, respectively. Bands that does not have any k-point with energy in the range [-2.0 eV, 2.0 eV] is not included in the calculation of the transport coefficients.

transport_include_bands

A list containing specific bands on which to calculate the transport coefficients. If the list is empty, use all bands within the range set by :ref:`transport_energycutband`. Band index starts at 1.

Example:

```
transport_include_bands: [3, 4, 10]
```

Calculate the transport coefficients for band 3, 4 and 10.

transport_use_analytic_scattering

Determines if the analytic parabolic scattering models should be used. They can be applied also to dispersions which are not parabolic, but such an application have to be physically justified.

Example:

```
transport_use_analytic_scattering: False
```

Use the density-of-states to set up the scattering mechanisms.

transport_drop_valence

Determines if all valence band should be dropped while reading e.g. external data. Currently only works for the VASP interface.

Example:

```
transport_drop_valence: False
```

Do not exclude the valence bands during read-in.

transport_drop_conduction

Determines if all conduction bands should be dropped while reading e.g. external data. Currently only works for the VASP interface.

Example:

```
transport_drop_conduction: False
```

Do not exclude the conduction bands during read-in.

Density of states

Here follows input parameters related to the calculation of the density of states.

dos_calc

Determines if the user wants to calculate the density of states. Even if this flag is set to *False*, the density of states is sometimes calculated if needed, e.g. if the density of states dependent scattering models are employed. However, with this parameter set to *True* and e.g. `transport_calc` set to *False* it is possible to only calculate the density of states.

```
dos_calc: False
```

Do not calculate the density of states.

dos_e_min

The minimum energy to use for the density of states calculation. In units of eV. The reference is with respect to the aligned Fermi level and consequetive shift that might have been applied. Note that the range of density of states calculation might change if it is called from other routines, e.g. the density of states dependent scattering models in order to cover enough energies.

```
dos_e_min: -5.0
```

Calculate the density of states from -5.0 eV.

dos_e_max

The maximum energy to use for the density of states calculation. In units of eV. The reference is with respect to the aligned Fermi level and consequetive shift that might have been applied. Note that the range of density of states calculation might change if it is called from other routines, e.g. the density of states dependent scattering models in order to cover enough energies.

```
dos_e_max: 2.0
```

Calculate the density of states to 2.0 eV.

dos_num_samples

The number of energy samples between `dos_e_min` and `dos_e_max`.

```
dos_num_samples: 1000
```

Use 1000 energy points from `dos_e_min` to `dos_e_max`.

dos_smearing

Gaussian smearing factor in units of eV. Only relevant if `dos_integrating_method` is set to *smeared*, *trapz*, *simps* or *romb*.

```
dos_smearing: 0.1
```

dos_integrating_method

Determines which method of integration to use to obtain the density of states. The following options are available:

- *trapz* - trapezoidal integration
- *simps* - Simpson integration
- *romb* - Romberg integration
- *tetra* - linear tetrahedron method without Blochl corrections

```
dos_integrating_method: "trapz"
```

Use trapezoidal integration to obtain the density of states.

General parameters

Here follows general parameters.

temperature_min

The minimum temperature in K.

Example:

```
temperature_min: 100
```

The minimum temperature is set at 100 K.

temperature_max

The maximum temperature in K.

Example:

```
temperature_max: 700
```

The maximum temperature is set at 700 K.

temperature_steps

The number of temperature steps from `temperature_min` to `temperature_max`.

Example:

```
temperature_steps: 7
```

In total 7 temperature steps, resulting in temperature samplings at 100, 200, 300, 400, 500, 600 and 700 K.

gamma_center

Gamma centered k-point grids? Anything else is currently not supported (or tested).

Example:

```
gamma_center: True
```

Notifies that the k-point grids are
Gamma centered.

maxeint

The limites of the dimensionless carrier energy
eta used for the numerical solution of the Fermi-Dirac integrals. Only relevant if `transport_method` is set to *numerick*.

Example:

```
maxeint: 100
```

Sets the limits of the Fermi-Dirac integrals to 100
eta.

occ_cutoff

The cutoff to use when detecting occupancies. Used for detecting the valence band maximum, conduction band minimum and then also for the band gap.

Example:

```
occ_cutoff: 1.0e-4
```

The occupancy cutoff is set at 1.0e-4, which means that states with an occupancy less than this will be assumed not occupied and vice versa.

e_fermi_in_gap

Determines if the Fermi level is to be placed in the middle of the gap.

Example:

```
e_fermi_in_gap: False
```

Do not place the Fermi level in the middle of the gap.

e_fermi

Determine if one should shift the energies to the supplied Fermi level (usually read in the interface).

Example:

```
e_fermi: True
```

Shift the energies such that zero is placed at the supplied Fermi level.

e_vbm

Determines if to set the Fermi level at the valence band maximum.

Example:

```
e_vbm: False
```

Do not set the Fermi level at the top valence band.

e_shift

After all alignments have been performed, perform this additional shift. Units in eV.

Example:

```
e_shift: 0.0
```

Sets the additional energy shift to 0 eV.

skw_expansion_factor

The expansion factor used in the SKW routine. It is basically tells how many unit cells that can be used. Only relevant if `dispersion_interpolate_method` is set to *skw*.

Example:

```
skw_expansion_factor: 5
```

Use 5 unit cells in each direction. In a second step a sphere is cut from this volume, thus removing the points in the far corners of this volume in the interpolation procedure.

carrier_valence_energy

The cutoff in which where to interpret the carriers as p-type. Used in the calculation of the carrier concentration. Units in eV.

Example:

```
carrier_valence_energy: 0.0
```

Would make sure all carriers at negative energies are interpreted as p-type.

carrier_conduction_energy

The cutoff in which where to interpret the carriers as n-type. Used in the calculation of the carrier concentration. Units in eV.

Example:

```
carrier_valence_energy: 0.0
```

Would make sure all carriers at positive energies are interpreted as n-type.

carrier_dos_analytick

Determines if the carrier concentration should be recaculated after being set up with analytical models. Only relevant if the band structure is generated from analytical models.

Example:

```
carrier_dos_analytick: True
```

Do not recalculate and use the analytical expressions for the carrier concentration.

defect_ionization

Determines if we shoudl use the expressions for the defect ionization in order to calculate the p- and n-type carrier concentration.

Example:

```
defect_ionization: False
```

Do not use the models for the defect_ionization to adjust the p- and n-type carrier concentration.

donor_number

The density of donors in units of $10^{-21}\mathrm{cm}^{-3}$.

Example:

```
donor_number: 0.0
```

No donors present.

donor_degen_fact

The degeneracy factor for the donors.

Example:

```
donor_degen_fact: 0.75
```

A degeneracy factor of 0.75 is used.

donor_energy

The energy of the donor in units of eV. Should be referenced to the energy after all adjustments to the Fermi level and additional energy shifts have been performed.

Example:

```
donor_energy: 0.0
```

The donor energy is 0 eV.

acceptor_number

The density of acceptors in units of $10^{-21}\mathrm{cm}^{-3}$.

Example:

```
donor_number: 0.0
```

No acceptors present.

acceptor_degen_fact

The degeneracy factor for the acceptors.

Example:

```
acceptor_degen_fact: 0.75
```

A degeneracy factor of 0.75 is used.

acceptor_energy

The energy of the acceptor in units of eV. Should be referenced to the energy after all adjustments to the Fermi level and additional energy shifts have been performed.

Example:

```
acceptor_energy: 0.0
```

The acceptor energy is 0 eV.

read

Determine how to set up the band structure and/or how to read data. The following options are possible:

- *param* - The band structure is generated from the parameter files. For all cases the band structure is generated by analytical models. The parameters pertaining to the construction of the bandstructure itself is set in the file `bandparam.yml`.
- *numpy* - Read data from NumPy datafiles without group velocities.

  The datastructure of the supplied numpy array
  should be on the following format:
  [
  [kx], [ky], [kz], [e_1], [v_x_1], [v_y_1], [v_z_1],
  [e_2], [v_x_2], [v_y_2], [v_z_2], ... ,
  [e_n], [v_x_n], [v_y_n], [v_z_n]
  ]

  The band parameters still need to be set in `bandparam.yml` as they contain necessary information about scattering etc.
- *numpyv* - Read data from NumPy datafiles, including group velocities.

  The datastructure of the supplied numpy array
  should be on the following format:
  [
  [kx], [ky], [kz], [e_1], [e_2], ... , [e_n]
  ]

  The band parameters still need to be set in `bandparam.yml` as they contain necessary information about scattering etc.
- *vasp* - Read data from a supplied VASP XML file, typically vasprun.xml. The band parameters still need to be set in `bandparam.yml` as they contain necessary information about scattering etc.

Example:

```
read: param
```

Construct the band structure from the parameters present in `bandparam.yml`.

readfile

The name of the file to be read. Depending on `read` it has the following behaviour:

- *param* - not relevant
- *vasp* - the name of the VASP XML file, if not set it defaults to *vasprun.xml*
- *numpy* - the name of the NumPy datafile
- *numpyv* - the name of the NumPy datafile

Example:

```
readfile: ""
```

Use defaults, e.g. vasprun.xml for VASP.

scissor

Apply a simple scissor operator to increase the band gap. Only works of the band gap has been correctly determined. In units of eV if not *False*.

Example:

```
scissor: False
```

Do not apply a scissor operator.

symprec

The symmetry cutoff parameters. Passed to Spglib. VASP also uses an internal symmetry parameter which is called *SYMPREC*. Spglib need to reproduce the symmetry that was detected in VASP in order for the k-point grids and thus the mapping between the IBZ and BZ to be valid. If errors regarding this is invoked, please try to adjust symprec.

Example:

```
symprec: 1.0e-6
```

If two coordinates are within 1.0e-6 it is assumed that they are the same and symmetry is thus detected.

libinfo

Determines if printout to stdout is performed in the interfaces to the external libraries.

Example:

```
libinfo: False
```

Do not print stdout information from the interfaces.

onlytotalrate

Determines if the users wants to store the relaxation time for each scattering mechanism. This is usefull for visualization purposes, but is simply very memory demanding. Users should try to leave this to *True*.

Example:

```
onlytotalrate: True
```

Only store the total relaxation time.

parallel

Determines if transport and density of states integrals are to be performed in parallel (embarrassingly). Currently this is not fully implemented, so users should leave this to *False*.

Example:

```
parallel: False
```

Do not use the parallel features.

run_tests

Determines if the tests are to be run. Several options are available:

- *slow* - Run all tests.
- *fast* - Only run the fast tests.
- *True* - Same as *fast*.
- *False* - Do not run any tests.

Example:

```
run_tests: False
```

Do not run any tests.

Band structure input parameters

```
    - tau0_c
    - emmission
```

Notes about format

The input files follow normal YAML conventions. Please inspect the sample file `input/bandparam.yml`. Even though many parameters have default values if not specified the user should always run the calculations with fully specified input files for consistency and reproducibility.

There is one entry per band. If many bands are used one can specify a range, e.g. Band X-Y: to set the same parameters for bands X to Y. Or if one would want to set the same parameters for all bands one should use Band 1-: This is quite usefull when reading data from a full-band calculation of some sort.

Remember to use two spaces indent after each Band entry (before a new Band entry) in order to comply with the YAML formatting standard.

Also, the parameters should be indented with two spaces from the Band entry:

Band 1-2:
  aparameter: somevalue
  anotherparameter: someothervalue

Band 3-5:
  aparameter: somevalue
  anotherparameter: someothervalue

Make sure all bands are specified. In this example, five bands was included.

General parameters

The following parameters are general and does not relate directly to a specific scattering mechanism etc.

`type`

Determines how to generate the bands if not read. Relevant only if `read` is set to *param*. The following options are possible:

- *0* - parabolic bands according to the relation

  where the effective mass m is set by `effmass`.

- *1* - parabolic bands pluss a quartic correction according to the relation:

  where the effective mass m is set by `effmass` and the correction factor a is set by `a`.

- *2* - Kane types (alpha correction) according to the relation:

  where the effective mass m is set by `effmass` and the correction factor $alpha$ is set by `a`.

No band folding is performed, except for the tight-binding case. It is important thus to scale the unit cell such that there is enough band coverage within the requested region of the chemical potentials pluss the excess needed for the thermal broadening.

effmass

The effective mass in units of the free electron mass along each configured unit vector in reciprocal cell. Use negative values to generate bands that curve down and vice versa.

Example:

```
effmass: [-1.0,-1.0,-1.0]
```

Generates band that for the parabolic case curves down with an effective mass along each unit vector of the configured recirprocal cell equal to the free electron mass.

a

The correction factor to be applied. See `type` for additional description. Is given along each unit vector in the configured reciprocal cell similar to the effective mass.

Example:

```
a: [-100.0,-100.0,-100.0]
```

Applies a correction factor of -100.0 along each unit vector direction in the currently configured reciprocal cell.

e0

An energy shift in units of eV. Applies to the current band.

Example:

```
e0: 0.0
```

Shift the band with 0.0 eV.

status

Determines if this is a valence or a conduction band. The following options are available:

- *v* - valence band
- *c* - conduction band

Example:

```
status: v
```

This band is a valence band.

kshift

Shift the band by a reciprocal vector, otherwise it is centered at Gamma. Have to be specified in cartesian coordinates.

Example:

```
kshift: [0.0,0.0,0.0]
```

Do not apply any shift to the current band.

spin_degen

The spin degeneracy of the current band. The following options are available:

- *1* - not spin degenerated
- *2* - spin degenerated

Example:

```
spin_degen: 2
```

The current band is spin degenerated.

General scattering related parameters

In the following the parameters related to the setup of the scattering mechanisms are given.

select_scattering

Determines which scattering mechnisms to apply for the current band. Set element to 1 to include scattering, 0 otherwise. Currently the following scattering mechanisms have been implemented (the number indicate array index, starting at 1):

- 1 elastic acoustic phonon scattering from def. pot.
- 2 non-polar optical phonon scattering
- 3 intervalley phonon scattering
- 4 polar optical phonon scattering
- 5 piezoelectric phonon scattering
- 6 ionized impurity scattering (Brooks-Herring)
- 7 ionized impority scattering (Conwell-Weiskopf)
- 8 alloy scattering
- 9-11 empty slots
- 12 constant scattering

If one does not use the analytic (parabolic) scattering models and instead use the density of states to generate the scattering rate, then only the first four and the last have been implemented (currently only the first and last have been properly tested)

Example:

```
select_scattering: [1,0,0,0,0,0,0,0,0,0,0,0]
```

Apply acoustic-phonon scattering by deformation potential to the following band.

explicit_prefact

Set an explicit prefactor for the relaxation time instead of using the prefactor from the density of states or parabolic band models. This behavior is enabled by setting the relevant element to *1* for the mechanism where one would like to specify an explicit prefact (constant tau0 is not included and is set below) for. Make sure that the total units that come out should be in fs. This is not always so easy to do due to temperature variations etc. Thus if the user also perform calculations at different temperatures, please consider that the prefactor usually change. This option should only be used by experts. If all elements in the array is *0*, the scattering models based on density of states or parabolic bands is used.

Example:

```
explicit_prefact: [0,0,0,0,0,0,0,0,0,0,0]
```

Disable the use of explicit prefactors.

explicit_prefact_values

The values of the explicit prefactors. Only relevant for the entries in `explicit_prefact` with a value of *1*. Remember that the units of the relaxation time come out as fs, including the energy dependency (density of states or parabolic band). Depending on the model, the prefactor thus have different units. Also consider that the prefactor usually has a temperature and effective mass dependence.

Example:

```
explicit_prefact_values: [0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
                          0.0, 0.0, 0.0, 0.0, 0.0]
```

All explicit prefactors of the relaxtion time is set to zero.

Acoustic phonon scattering parameters

This model uses the linear Debye model.

d_a

Acoustic deformation potential in units of eV. Remember to rescale this is the overlap matrix is not one.

Example:

```
d_a: 10
```

Use a deformation potential of 10 eV.

speed_sound

The speed of sound. This is the group velocity of the low energy acoustic branch that is in the Debye model assumed to be linear. In units of m/s.

Example:

```
speed_sound: 10000
```

Use a group velocity of 10000 m/s.

Piezoelectric phonon scattering parameters

This model uses the polarization that is set up due to strain effects to describes acoustic phonon scattering. Typically important for polar materials.

p

The piezoelectric constant in units of C/m^2

Example:

```
p: 0.0
```

The piezoelectric constant is set to zero.

isl

The inverse screening length in the Debye formulation in units of inverse AA.

Example:

```
isl: 0.0
```

The inverse screening length is set to zero.

Non-polar optical phonon scattering

This model uses the Einstein model of a optical phonon mode (dispersion assumed to be flat so a constant value is used for the frequency).

d_o

The optical deformation potential in units of eV/AA.

Example:

```
d_o: 35.0
```

The optical deformation potential is set to 35.0 eV/AA.

n_o

The occupation number of the optical phonon.

Example:

```
n_o: 0.0
```

The occupation number of the optical phonon is set to zero.

omega_o

The optical phonon frequency to use from the Einstein model. In units of THz.

Example:

```
omega_o: 0.0
```

The optical phonon frequency is set to zero.

Polar optical phonon scattering

After the Froelich model. Should be replaced for a more explicit model in the future.

epsi

The permitivity of the electron in units of the vacuum permitivity.

Example:

```
epsi: 0.0
```

The permitivity is set to zero.

f

The Froehlich term.

Example:

```
f: 0.0
```

The Froechlich term is set to zero.

Intervalley acoustic phonon scattering

A model where the electron scatters both of acoustic and optical phonon modes. E.g. phonons connect two valleys.

n_vv

The intervalley phonon occupation number.

Example:

```
n_vv: 0.0
```

The intervalley phonon occupation number is set to zero.

omega_vv

The transition frequency in units of THz.

Example:

```
omega_vv: 0.0
```

The transition frequency is set to zero.

etrans

The transition energy between the bottom of the two values. In units of eV.

Example:

```
etrans: 0.0
```

The transition energy is set to zero.

zf

The number of possible final states (final state degeneracy).

Example:

```
zf: 0.0
```

The number of final states is set to zero.

q_energy_trans

The scattering vector connecting the two valleys in direct reciprocal coordinates.

Example:

```
q_energy_trans: [[0,0,0],[0.5,0.5,0.5]]
```

The scattering vector is set along the diagonal reciprocal cell.

Ionized impurity scattering parameters

Parameters using either the Conwell and Weisskopf (CW) or the Broks and Herring (BH) model to describe ionized impurity scattering.

n_i

The density of ionized impurities in units of $10^{21}\mathrm{cm}^{-3}$. Used for both the CW and BH model.

Example:

```
n_i: 0.01
```

The density of ionized impurities is set to $10^{19}\mathrm{cm}^{-3}$.

isl_i

The inverse screening length in units of inverse AA. Only used for the BH model.

Example:

```
isl_i: 0.3
```

The inverse screening length is set to 0.3 inverse AA.

z

The number of charge units of the impurity. In units of the electron charge.

Example:

```
z: 1.0
```

The charge of the impurity is set to one electron charge.

Alloy scattering parameters

A scattering model for the alloy $\mathrm{A}_x\mathrm{B}_{1-x}\mathrm{C}$.

vdiff

The atomic potential difference between the species A and B in eV.

Example:

```
vdiff: 1.0
```

The potential difference is set to 1.0 eV.

alloyconc

The concentration, x of the alloy.

Example:

```
alloyconc: 0.5
```

The concentration is set to 50%, i.e. 50% of A and 50% of B.

Common scattering parameters

Here follows scattering parameters that are shared between the different scattering mechnisms.

eps

The dielectric constant in units of the vacuum value.

Example:

```
eps: 12.0
```

The dielectric constant is set to 12.0 times the vacuum value.

rho

The mass density of the material in $\mathrm{g/cm^3}$.

Example:

```
rho: 2.4
```

The mass density of the material is set to 2.4 $\mathrm{g/cm^3}$.

tau0_c

The value of the constant relaxation time in units of fs.

Example:

```
tau0_c: 100.0
```

The constant relaxation time is set at 100.0 fs.

emmission

Determines if the considered scattering mechnism is by emmision or absorption. Acoustic phonon scattering includes both so this is only relevant where scattering of optical phonons is encountered.

Example:

```
emission: False
```

Use absorption, i.e. a phonon is absorbed in the scattering event.

Unit cell input parameters

Notes about format

The input files follow normal YAML conventions. Please inspect the sample file `input/cellparam.yml`. Even though many parameters have default values if not specified the user should always run the calculations with fully specified input files for consistency and reproducibility.

The unit cell

a

The first lattice vector describing the unit cell in AA.

b

The second lattice vector describing the unit cell in AA.

c

The third lattice vector describing the unit cell in AA.

Example:

```
a: [5.0,0.0,0.0]
b: [0.0,5.0,0.0]
c: [0.0,0.0,5.0]
```

Generates a unit cell that is 5.0 by 5.0 by 5.0 AA.

Atomic positions

direct

Determines if the atomic positions are given in direct coordinates.

Example:

```
direct: True
```

The atomic positions are given in direct coordinates

pos

A list of the atomic positions. In direct or cartersian coordinates depending on the parameter `direct`.

Example:

```
pos: [[0.0,0.0,0.0]]
```

One atom centered at origo.

atomtypes

The type of atoms as a list in the same order as `pos`. Use abbreviations that are standard to the periodic table. An addition element X is added for unknown types.

Example:

```
atomtypes: [X]
```

K-point grid density

ksampling

The k-point sampling along each axis of the configured reciprocal unit cell.

Example:

```
ksampling: [15,15,15]
```

Use 15 by 15 by 15 samples in the full reciprocal unit cell.

6.6 Output

The transport coefficients are written to the *output* directory. The log file `info.log` is also written in this directory and can be monitored during a run to check the process.

6.6.1 Files

Electrical conductivity

The electrical conductivity can be found in the `sigma`. In units of $\mathrm{S/m}$. Consult the documentation in the header of the output file for layout.

Seebeck coefficients

The Seebeck coefficient can be found in the `seebeck`. In units of $\mu\mathrm{V/K}$. Consult the documentation in the header of the output file for layout.

Lorenz coefficient

The Lorenz coefficient can be found in the `lorenz`. In units of $10^{-8}\mathrm{V}^2/\mathrm{K}^2$. Consult the documentation in the header of the output file for layout.

The electrothermal conductivity

The electrical part of the thermal conductivity can be found in the `kappae`. In units of $\mathrm{W/mK}$.

The charge carrier concentration

The charge carrier concentration can be located in the `cc`. In units of $10^{21}\mathrm{cm}^{-3}$. Consult the documentation in the header of the output file for layout.

The Hall coefficient

The Hall coefficient (big R) can be located in the `hall`. In units of cm^3/C. Consult the documentation in the header of the output file for layout.

Warning: NOT YET IMPLEMENTED WHEN FIRST-PRINCIPLE INPUT IS UTILIZED (ONLY FILLED WITH BOGUS DATA). ONLY WORKS FOR SPHERICAL BANDS AT THE MOMENT.

The relaxation times

The total relaxation time for each band can be found in the files `scattering_band_n`, for band number *n*. In units of fs.

6.6.2 Visualization

Data can easily be visualized with Gnuplot. Currently no automatic visualization is performed. Data is blocked on temperature.

6.7 Tutorials

This document is currently empty, please consult the examples while the content is developed.

6.8 Examples

6.8.1 Silicon from first-principles

Here a short example of how to calculate the transport coefficients from a VASP output file (typically vasprun.xml) is presented.

Preparation

Start with the `param.yml`, `bandparam.yml` located in the *tests/13* directory. In the same directory a sample `vasprun.xml` file is also provided.

Copy these files into an *input* directory in the directory where you want to execute T4ME.

The general parameters

Here follows a brief explanation of the general parameters that need attention. They are already specified in the sample `param.yml` file so the user should not have to change this for a test run. All other parameters should at this point not need to be touched.

```
dispersion_interpolate: False
```

We do not want to interpolate the input at this step.

```
dispersion_velocities_numdiff: True
```

We need to calculate the group velocities of the electrons as this is not supplied by VASP by default.

```
transport_calc: True
```

Calculate the transport coefficients.

```
transport_method: "numerick"
```

Integrate numerically in k-space.

```
transport_integration_method: "trapz"
```

Used trapezoidal integration.

```
transport_chempot_min: -0.4
```

The minimum chemical potential to sample, in eV.

```
transport_chempot_max: 1.0
```

The maximum chemical potential to sample, in eV.

```
transport_chempot_samples: 20
```

How many samples do you want between `transport_chempot_min` and `transport_chempot_max`. Computational time and storage may vary depending on how this parameter is set.

```
transport_use_analytic_scattering: False
```

Use density-of-states models for the electron scattering.

```
dos_integrating_method: "trapz"
```

In order to calculate the scattering, it is necessary to calculate the density-of-states. Here this is done by using the trapezoidal integration with the delta function approximated by a Gaussian.

```
dos_smearing: 0.1
```

The smearing factor in eV used for the Gaussian approximation of the delta function. This needs to be sufficiently big in order for the density-of-states to converge, but also in order for the scattering data to have a smooth onset at the carrier energy. It is recommended that this, for calculations of scattering properties is not set below 0.1 eV.

```
temperature_min: 300
```

The minimum temperature in K.

```
temperature_max: 300
```

The maximum temperature in K.

```
temperature_steps: 1
```

The number of temperature steps between `temperature_min` and `temperature_max`.

```
e_fermi: True
```

Set the zero in energy to Fermi level supplied by the first-principle code, here VASP. The `transport_chempot_min` and `transport_chempot_max` parameters are thus set with reference to the shifted grid where the zero in energy is usually at the top valence band.

```
read: vasp
```

Read data from VASP output files, here `vasprun.xml`.

```
symprec: 1e-6
```

The symmetry cutoff used to detect symmetry. Should somewhat match with the value used in VASP. Passed along to Spglib to generate the irreducible to full Brillouin zone mapping.

```
onlytotalrate: True
```

Only store the total concatenated relaxation time arrays. Saves memory.

The band parameters

Here follows a brief explanation of the band parameters that need attention. They are already specified in the sample `bandparam.yml` file so the user should not have to change this for a test run. All other parameters should at this point not need to be touched.

```
Band 1-:
```

Tells the reader that it should apply all consecutive parameters to all the bands in the supplied system.

```
select_scattering: [0,0,0,0,0,0,0,0,0,0,0,1]
```

Only use constant scattering.

```
tau0_c: 100
```

The value of the constant relaxation time in fs.

Execution

After all parameters have been set (should only be necessary to copy files as stated before) the transport coefficients can be calculated by executing

```
python t4me.py
```

During execution the file `info.log` in the directory *output* can be inspected in order to assess progress and that everything works as expected.

Output

On completion the transport coefficients can be found in the *output* directory.

6.9 Custom interfaces

Here comes a description of how to customize new interfaces. In the meantime, please consult the structure and content of the `interfaces.py` file which among other contains interfaces to VASP which sets up the necessary data structure. This can be copied and modified to work against other codes.

6.10 API Documentation

6.10.1 scattering module

This module, `scattering.py` contains the setup and calculation of the scattering properties.

6.10.2 t4me module

This module, `t4me.py` contains the main driver for the program. Most standard calls are included, but if the users wants, custom execution can be adapted in this file.

It is with this file T4ME is executed with the command

```
python t4me.py
```

6.10.3 transport module

This module, `transport.py` contains the setup and checks necessary to execute the calculation of the transport coefficients. It also execute those routines.

6.10.4 lbteint module

This module, `lbteint.py` contains the integral solvers for the linearized Boltzmann Transport equations.

6.10.5 lbtecoeff module

This module, `lbtecoeff.py` contains the setup and calls to the integral solvers and possible also external integral solvers such that the linearlized Boltzmann Transport Equations might be solved.

6.10.6 lattice module

This module, `lattice.py` contains the setup and lattice related routines, including both the real and reciprocal part of it.

6.10.7 bandstructure module

This module, `bandstructure.py` contains the setup and calculation of bandstructure related properties. This also include for instance the calculation of the density of states.

6.10.8 inputoutput module

This module, `inputoutput.py` handles input and output related acitivites.

6.10.9 interface module

This module, `interface.py` contains the interfaces which reads in data and calculates for instance parametrized bandstructures etc.

6.10.10 utils module

This module, `utils.py` contains general routines.