effmass Documentation

Release 2.3.2.dev3

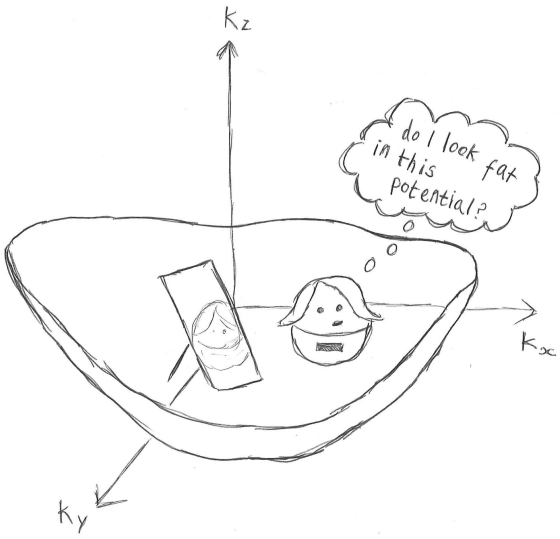
Lucy Whalley

Sep 13, 2023

Contents

effmass is a Python package for calculating various definitions of effective mass from the electronic bandstructure of a semiconducting material. It consists of a core class that calculates the effective mass and other associated properties of selected bandstructure segments. The module also contains functions for locating bandstructure extrema and plotting approximations to the dispersion.

Examples are provided in a Jupyter notebook tutorial here. Source code is available as a git repository at https://github.com/lucydot/effmass.



CHAPTER 1

Details

Authors Lucy Whalley

Contact l.whalley@northumbria.ac.uk

GitHub https://github.com/lucydot

Citation `bibtex`

CHAPTER 2

Documentation

2.1 Features

effmass can:

Read in a bandstructure: It is assumed you have used a DFT calculator to walk through a 1D slice of the Brillouin Zone, capturing the maxima and minima of interest. *effmass* uses the Python packages [vasppy](https://github.com/bjmorgan/vasppy) for parsing *VASP* output, and *ASE* for parsing the *Castep* output.

Locate extrema: These correspond to the valence band maxima and conduction band minima. Maxima and minima within a certain energy range can also be located.

Calculate curvature, transport and optical effective masses: The curvature (aka inertial) and transport masses are calculated using the derivatives of a fitted polynomial function. The optical effective mass can also be calculated assuming a Kane dispersion.

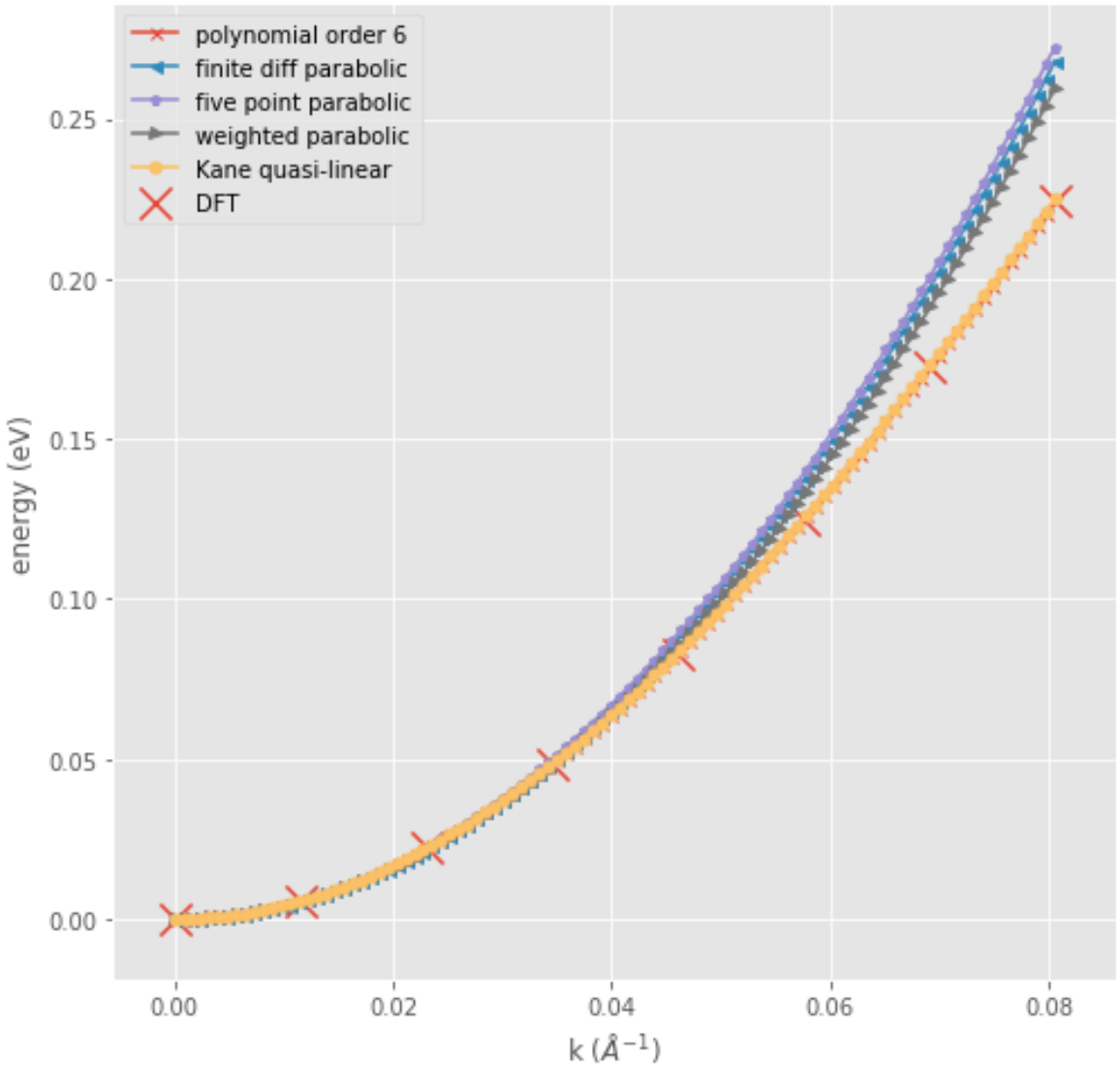
Assess the extent of non-parabolicity: Parameters of the Kane quasi-linear dispersion are calculated to quantify the extent of non-parabolicity over a given energy range.

Calculate the quasi-fermi level for a given carrier concentration: Using density-of-states data and assuming no thermal smearing, *effmass* can calculate the energy to which states are occupied. This is a useful approximation to the quasi-Fermi level. *Note: this is only supported for VASP and requires the output file 'DOSCAR'.*

Plot fits to the dispersion: Selected bandstructure segments and approximations to the dispersion (assuming a Kane, quadratic, or higher order fit) can be visualised.

The command line interface provides basic functionality for calculating parabolic effective masses. For those who have a basic familiarity with Python there is an API which provides access to more (non-parabolic) effective mass definitions. Depending on the functionality and level of approximation you are looking for, it may be that one of the packages listed [here](https://effmass.readthedocs.io/en/latest/Related%20packages.html) will suit your needs better.

```
electron [1 1 1]
finite difference mass is 0.09
3-point parabolic mass is 0.09
weighted parabolic mass is 0.10
alpha is 1.021 1/eV
kane mass at bandedge is 0.089
range of quasi-linear until 0.224 eV
optical mass at band edge (assuming Kane dispersion) is 0.10
```



2.2 Supported Codes

effmass currently supports *VASP*, *FHI-Aims*, *Castep* and *ASE*. In the near future we hope to play nicely with other codes that interface with the ASE bandstructure class, and pymatgen. We especially welcome contributions that will help make *effmass* available to more researchers.

2.3 Installation

effmass is a Python 3 package and requires key packages from the SciPy ecosystem: SciPy, NumPy and Matplotlib. If you have not installed these packages before, it may be best to install them using your preferred package manager (eg: Homebrew). Note that together they will use >100MB of disk space. *effmass* can then be installed using the Python package manager *pip*:

```
pip install effmass
```

To start the command line interface simply type

```
effmass
```

2.3.1 Alternative Installation Methods

If you use conda/anaconda, the safest thing to do is to create a new environment and then install *effmass* and all of its dependencies (which will use >300MB of disk space):

```
conda create -n effmass python
conda activate effmass
pip install effmass
```

If you wish, you can install the very latest version of *effmass* from GitHub with the commands below. **Note**: The latest GitHub version may include more features and data format support that the latest release, but it is not a stable release, so may have more issues than usual. If you are unsure, use one of the above install methods instead.

```
git clone https://github.com/lucydot/effmass.git
# or git clone git@github.com:lucydot/effmass.git
cd effmass
pip install .
```

2.4 API documentation

2.4.1 effmass.inputs

class effmass.inputs.**Data**

Parent class for parsing and storing data from bandstructure calculations. Contains a *check_data()* method for basic checks on bandstructure data.

spin_channels

1 (non-spin-polarised), 2 (spin-polarised), 4 (spin-orbit coupling).

Type int

number_of_kpoints
the number of k-points per band.

Type int

number_of_bands
the number of bands.

Type int

kpoints
2-dimensional array with shape (number_of_kpoints, 3). Each row contains the fractional coordinates of a kpoint [kx,ky,kz].

Type array(float)

energies
2-dimensional array with shape (number_of_bands,number_of_kpoints). Each row contains energies of eigenstates in eV for a particular band.

Type array(float)

occupancy
2-dimensional array with shape (number_of_bands,number_of_kpoints). Each row contains occupation number of the eigenstates for a particular band. Values range from 0-1 (spin-polarised) or 0-2 (non-spin-polarised).

Type array(float)

reciprocal_lattice
the reciprocal lattice vectors in format [[x1,y1,z1],[x2,y2,z2],[x3,y3,z3]], units Angstrom $^{-1}$.

Type list(float)

CBM
the conduction band minimum energy in eV.

Type float

VBM
the valence band maximum in eV.

Type float

fermi_energy
the fermi energy in eV.

Type float

__init__()
Initialises an instance of the *Data* class. All attributes are None until set by the derived class.

Parameters **None.** –

Returns None.

check_data(*spin_channels*, *number_of_kpoints*, *number_of_bands*, *CBM*, *VBM*, *fermi_energy*, *occupancy*)
Check that Data class attributes make basic sense.

Parameters **None.** –

Returns None.

Notes

There is a similar method that runs automatically when reading data in using the vasppy.procar module.

class effmass.inputs.**DataASE**(*bs*, *atoms*)

Class for interfacing with the ASE bandstructure object. Inherits attributes and methods from the *Data* class, and extends with a method for inferring the CBM/VBM from Fermi level.

Note: DataASE.fermi_energy is taken from the seedname.out file.

Note: The DataASE class does not parse eigenstate occupancy data. The Fermi energy will be used to infer which bands are occupied (below the fermi energy) and which are unoccupied (above the fermi energy). You should independently confirm that the fermi energy is in the band gap of your material. Note that you can manually set the *fermi_energy* attribute and find the CBM and VBM using the method *find_cbm_vbm*. “)

__init__(*bs*, *atoms*)

Initialises an instance of the *DataASE* class and infers which bands are occupied and unoccupied from the fermi level.

Parameters **bs** (*ase.spectrum.band_structure.BandStructure*) – An instance of the ase.spectrum.band_structure.BandStructure object.

Returns None.

class effmass.inputs.**DataAims**(*directory_path*, *output_name='calculation.out'*)

Class for parsing and storing data from a FHI-AIMS calculation.

spin_channels

1 (non-spin-polarised), 2 (spin-polarised), 4 (spin-orbit coupling).

Type int

number_of_kpoints

the number of k-points per band.

Type int

number_of_bands

the number of bands.

Type int

kpoints

2-dimensional array with shape (number_of_kpoints, 3). Each row contains the fractional coordinates of a kpoint [kx,ky,kz].

Type array(float)

energies

2-dimensional array with shape (number_of_bands,number_of_kpoints). Each row contains energies of eigenstates in eV for a particular band.

Type array(float)

occupancy

2-dimensional array with shape (number_of_bands,number_of_kpoints). Each row contains occupation number of the eigenstates for a particular band. Values range from 0-1 (spin-polarised) or 0-2 (non-spin-polarised).

Type array(float)

reciprocal_lattice

the reciprocal lattice vectors in format [[x1,y1,z1],[x2,y2,z2],[x3,y3,z3]], units Angstrom $^{-1}$.

Type list(float)

CBM
the conduction band minimum energy in eV.

Type float

VBM
the valence band maximum in eV.

Type float

fermi_energy
the fermi energy in eV. Automatically set to the mean of Data.CBM and Data.VBM.

Type float

__init__(*directory_path*, *output_name='calculation.out'*)
Initialises an instance of the *DataAims* class and checks data using `check_data()`.

Parameters

- **directory_path** (*str*) – The path to the directory containing output, geometry.in, control.in and bandstructure files
- **output_name** (*str*) – Name of the output file - contrary to the rest of the files, this is chosen by the user during an Aims run. Defaults to 'aims.out'.

Returns None.

fermi_energy = None
Cutting energy values in a range of 30 eV above and below the Fermi level. FHI AIMS is all electron, but not all states are needed for a meaningful effmass calculation

number_of_bands = None
Finding number of kpoints and determining number of BZ paths

reciprocal_lattice = None
Finding spin channels

spin_channels = None
Finding number of bands

class effmass.inputs.**DataCastep**(*directory_path*, *seedname*)
Class for parsing and storing data from a Castep bandstructure calculation. Inherits attributes and methods from the *DataASE* class.

__init__(*directory_path*, *seedname*)
Initialises an instance of the *DataCastep* class.

Parameters

- **directory_path** (*str*) – The path to a directory containing seedname.cell, seedname.out and seedname.bands
- **seedname** (*str*) – The name (without suffix) of the input and output files

Returns None.

class effmass.inputs.**DataOctopus**(*bandstructure_path*, *info_path*, *results_path*)
Class for parsing and storing data from a octopus calculation. Extends the *Data* class"

Note: DataOctopus.fermi_energy is automatically set to the mean of DataOctopus.CBM and DataOctopus.VBM.

__init__(*bandstructure_path*, *info_path*, *results_path*)
Initialises an instance of the *Data* class and checks data using `check_data()`.

Parameters

- **bandStructure_path** (*str*) – The path to the bandstructure file.
- **info_path** (*str*) – The path to the info file.
- **results_path** (*str*) – The path to the results.out file.

Returns None.

class effmass.inputs.**DataVasp**(*outcar_path*, *procar_path*, *ignore=0*, ***kwargs*)
Class for parsing and storing data from a vasp calculation. Extends the *Data* class to include support for analysing DOSCAR data"

Additional attributes: dos (array): 2-dimensional array. Each row contains density of states data (units "number of states / unit cell") at a given energy: [energy(float),dos(float)]. integrated_dos: 2-dimensional array. Each row contains integrated density of states data at a given energy: [energy(float),integrated_dos(float)].

Note: DataVasp.fermi_energy is automatically set to the mean of DataVasp.CBM and DataVasp.VBM.

__init__(*outcar_path*, *procar_path*, *ignore=0*, ***kwargs*)
Initialises an instance of the *Data* class and checks data using `check_data()`.

Parameters

- **outcar_path** (*str*) – The path to the OUTCAR file
- **procar_path** (`str` or `list`) – The path(s) to one or more PROCAR files.
- **ignore** (*int*) – The number of kpoints to ignore at the beginning of the bandstructure slice through kspace (useful for hybrid calculations where zero weightings are appended to a previous self-consistent calculation).
- ****kwargs** – Additional keyword arguments for reading the PROCAR file(s).

Returns None.

parse_DOSCAR(*filename='./DOSCAR'*)
Parses the DOS and integrated DOS from a vasp DOSCAR file.

Parameters **filename** (*str, optional*) – The location and filename of the DOSCAR to read in. Defaults to *'./DOSCAR'*.

Returns None.

Notes

If the DOS has been sampled at more than 10000 points then this function will break at the expression for *num_data_points*. In this case, edit your DOSCAR file so that in the header there is a space preceding the number of points.

class effmass.inputs.**DataVasprun**(*path*)
Class for parsing and storing data from a VASP calculation using vasprun.xml. Works for parsing calculations with split k-point paths

Note: occupancies are set to 0 below fermi level and 1 above it

__init__(*path*)
Initialises an instance of the *Data* class and checks data using `check_data()`.

Parameters

- **path** (*str*) – Path to vasprun.xml. If the calculation was split along
- **k-path, the path should be to the folder which contains the** (*the*) –
- **i.e. for mapi/split-01/vasprun.xml, mapi/split-02/vasprun.xml** (*splits.*) –
- **would specify path=mapi** (*you*) –

Returns None.

class effmass.inputs.**Settings**(*energy_range=0.25*, *extrema_search_depth=0.025*, *conduction_band=True*, *valence_band=True*, *direction=None*, *frontier_bands_only=False*, *bandfit=6*, *degeneracy_condition=1e-05*)

Class for setting analysis parameters.

energy_range
energy in kT over which the segment extends.

Type float

extrema_search_depth
energy in kT from bandedge over which to search for extrema.

Type float

conduction_band
calculate conduction band (electron) effective masses. Defaults to True.

Type bool

valence_band
calculate valence band (hole) effective masses. Defaults to True.

Type bool

direction
calculate effective masses for this direction only. If None then effective masses for all directions are calculated. Defaults to False.

Type list(float)

frontier_bands_only
calculate effective masses for the lowest energy conduction band and/or highest energy valence band only. When True this overrides *extrema_search_depth*. Defaults to False.

degree_bandfit
the degree of the polynomial which is used to fit to dispersion data when calculating the transport mass.

Type int

degeneracy_condition
the energy difference below which bands are considered to be degenerate. Defaults to 1E-5.

Type float

__init__(*energy_range=0.25*, *extrema_search_depth=0.025*, *conduction_band=True*, *valence_band=True*, *direction=None*, *frontier_bands_only=False*, *bandfit=6*, *degeneracy_condition=1e-05*)

Initialises an instance of the Settings class and checks input using *check_settings()*.

Parameters

- **energy_range** (*float*) – energy in eV over which the segment extends. Defaults to 0.25 eV.
- **extrema_search_depth** (*float*) – energy in eV from bandedge over which to search for extrema. Defaults to 0.025 eV.
- **conduction_band** (*bool*) – calculate conduction band (electron) effective masses. Defaults to True.
- **valence_band** (*bool*) – calculate valence band (hole) effective masses. Defaults to True.
- **direction** (*list(float or int)*) – calculate effective masses for this direction only. If None then effective masses for all directions are calculated. Defaults to False.
- **frontier_bands_only** – calculate effective masses for the lowest energy conduction band and/or highest energy valence band only. When True this overrides *extrema_search_depth*. Defaults to False.
- **bandfit** (*int*) – the degree of the polynomial which is used to fit to dispersion data when calculating the transport mass.
- **degeneracy_condition** (*float*) – the energy difference below which bands are considered to be degenerate. Defaults to 1E-5.

Returns None.

check_settings()

Check that Settings class attributes are sane.

Parameters **None.** –

Returns None.

2.4.2 effmass.extrema

A module for finding the band structure extrema and instantiating a `Segment` object for each extrema point.

The extrema are found within an energy range given by the `Settings` class. Each *Segment* object contains data for kpoints within an energy range given by the `Settings` class.

effmass.extrema.**calc_CBM_VBM_from_Fermi**(*Data*, *CBMVBM_search_depth=4.0*)

This function is used to find the CBM and VBM when there is no occupancy data. It relies upon the Fermi level being in the middle of the band gap. The CBMVBM_search_depth is refereced from the fermi energy.

Parameters **DataASE** (DataASE) – instance of the `DataASE` class.

Returns A tuple containing the conduction band minimum and valence band maximum in eV.

Return type (float, float)

effmass.extrema.**calculate_direction**(*a*, *b*)

Calculates the direction vector between two points.

Parameters

- **a** (*list*) – the position vector of point a.
- **b** (*list*) – the position vector of point b.

Returns The (unnormalised) direction vector between points a and b. The smallest magnitude of an element is 1 (eg: [1,1,2]).

Return type array

effmass.extrema.**change_direction**(*kpoints*, *kpoint_indices*)
Finds the index of the kpoint (if any) where there is a change of direction in reciprocal space.

Parameters

- **kpoints** (*array*) – array of kpoints with shape (number_of_kpoints, 3). Each row contains the fractional coordinates of a kpoint [kx,ky,kz]. See *effmass.inputs.Data.kpoints*.
- **kpoint_indices** (*list (int)*) – the kpoint indices over which to check for change in direction

Returns the index of the kpoint where there is a change of direction. If there is no change of direction, returns None.

Return type int

effmass.extrema.**filter_segments_by_direction**(*segment_list*, *direction*)
Filter a list of Segments so that only those in a particular direction remain.

Parameters

- **segment_list** (*list(Segment)*) – A list of Segment objects.
- **direction** (*array(float)*) – The direction array, length 3.

Returns A list of Segment objects.

Return type segment_list (list(*Segment*))

effmass.extrema.**find_CB_indices**(*Data*, *Settings*)
Finds the kpoint index and band index of the minimum energy turning points within *effmass.inputs.Settings.energy_range* of the conduction band minimum (*effmass.inputs.Data.CBM*). Return indices for the lowest energy CB only if *frontier_bands_only* is True.

Parameters

- **Data** (Data) – instance of the Data class.
- **Settings** (Settings) – instance of the Settings class.

Returns A 2-dimensional array. Contains [efmmas.inputs.Data.bands index, *effmass.inputs.Data.kpoints* index] for each minima.

Return type array

effmass.extrema.**find_VB_indices**(*Data*, *Settings*)
Finds the kpoint index and band index of the maximum energy turning points within *effmass.inputs.Settings.energy_range* of the valence band maximum (*effmass.inputs.Data.VBM*). Return indices for the highest energy VB only if *frontier_bands_only* is True.

Parameters

- **Data** (Data) – instance of the Data class.
- **Settings** (Settings) – instance of the Settings class.

Returns A 2-dimensional array. Contains [efmmas.inputs.Data.bands index, *effmass.inputs.Data.kpoints* index] for each maxima.

Return type array

effmass.extrema.**generate_segments**(*Settings*, *Data*, *bk=None*, *truncate_dir_change=True*)
Generates a list of Segment objects.

Args: Settings (Settings): instance of the `Settings` class. Data (Data): instance of the `Data` class. truncate_dir_change (bool): If True, truncates eigenstates when there is a change in direction. If False, there is no truncation. Defaults to True. bk (list(int)): To manually set an extrema point, in format [*effmass.inputs.Data.energies* row index, *effmass.inputs.Data.kpoints* row index]. Defaults to None.

Returns A list of `Segment` objects.

Return type list(*Segment*)

effmass.extrema.**get_frontier_CB_indices**(*Data*, *CB_min_indices*, *degeneracy_condition*)

Returns the indices of the lowest energy minima across the Brillouin Zone

Parameters

- **Data** (`Data`) – instance of the `Data` class.
- **CB_min_indices** (*array(int)*) – A 2-dimensional array. Each row contains [`efmmas.inputs.Data.bands` index, *effmass.inputs.Data.kpoints* index] for each minimum in the CB band.

Returns A 2-dimensional array. Each row contains [`efmmas.inputs.Data.bands` index, *effmass.inputs.Data.kpoints* index] for each minimum in the frontier conduction band(s).

Return type frontier_indices (array(int))

effmass.extrema.**get_frontier_VB_indices**(*Data*, *VB_max_indices*, *degeneracy_condition*)

Returns the indices of the highest energy maxima across the Brillouin Zone

Parameters

- **Data** (`Data`) – instance of the `Data` class.
- **VB_max_indices** (*array(int)*) – A 2-dimensional array. Each row contains [`efmmas.inputs.Data.bands` index, *effmass.inputs.Data.kpoints* index] for each maximum in the VB band.

Returns A 2-dimensional array. Each row contains [`efmmas.inputs.Data.bands` index, *effmass.inputs.Data.kpoints* index] for each maximum in the frontier valence band(s).

Return type frontier_indices (array(int))

effmass.extrema.**get_kpoints_after**(*band_index*, *kpoint_index*, *Settings*, *Data*, *truncate_dir_change=True*)

For a given eigenstate, finds eigenstates which 1) belong to the same band 2) come after the given eigenstate in the route through reciprocal space 3) are within *effmass.inputs.Settings.energy_range*.

Parameters

- **band_index** (*int*) – index of `effmass.inputs.Data.bands`.
- **kpoint_index** (*int*) – index of *effmass.inputs.Data.kpoints*.
- **Settings** (`Settings`) – instance of the `Settings` class.
- **Data** (`Data`) – instance of the `Data` class.
- **truncate_dir_change** (*bool*) – If True, truncates eigenstates when there is a change in direction. If False, there is no truncation. Defaults to True.

Returns indices of *effmass.inputs.Data.kpoints*.

Return type list(int)

effmass.extrema.**get_kpoints_before**(*band_index*, *kpoint_index*, *Settings*, *Data*, *truncate_dir_change=True*)

For a given eigenstate, finds eigenstates which 1) belong to the same band 2) come before the given eigenstate in the route through reciprocal space 3) are within *effmass.inputs.Settings.energy_range*.

Parameters

- **band_index** (*int*) – index of effmass.inputs.Data.bands.
- **kpoint_index** (*int*) – index of *effmass.inputs.Data.kpoints*.
- **Settings** (Settings) – instance of the Settings class.
- **Data** (Data) – instance of the Data class.
- **truncate_dir_change** (*bool*) – If True, truncates eigenstates when there is a change in direction. If False, there is no truncation. Defaults to True.

Returns indices of *effmass.inputs.Data.kpoints*.

Return type list(int)

effmass.extrema.**get_maximum_indices**(*Data*, *extrema_search_depth*)

Finds the kpoint indices and band indices of all maximum turning points in VB within *extrema_search_depth*.

Parameters

- **Data** (Data) – instance of the Data class.
- **extrema_search_depth** (*float*) – energy in kT from bandedge over which to search for maxima.

Returns A 2-dimensional array. Each row contains [efmmas.inputs.Data.bands index, *effmass.inputs.Data.kpoints* index] for each maximum in the VB band.

Return type array

effmass.extrema.**get_minimum_indices**(*Data*, *extrema_search_depth*)

Finds the kpoint indices and band indices of all minimum turning points in CB within *extrema_search_depth*.

Parameters

- **Data** (Data) – instance of the Data class.
- **extrema_search_depth** (*float*) – energy in kT from bandedge over which to search for minima.

Returns A 2-dimensional array. Each row contains [efmmas.inputs.Data.bands index, *effmass.inputs.Data.kpoints* index] for each minimum in the CB band.

Return type array

A module for finding the band structure extrema and instantiating a Segment object for each extrema point.

The extrema are found within an energy range given by the Settings class. Each *Segment* object contains data for kpoints within an energy range given by the Settings class.

2.4.3 effmass.dos

A module for analysing DOSCAR data.

effmass.dos.**electron_fill_level**(*Data*, *volume*, *concentration*, *CBM_index*)

Finds the energy to which a given electron concentration will fill the density of states in integrated_dos.

Uses linear interpolation to estimate the energy between two points given in the DOSCAR.

Parameters

- **Data** (Data) – Instance of the Data class.
- **volume** (*float*) – volume of the unit cell in angstrom ^{3}.
- **concentration** (*float*) – electron concentration in cm $^{-3}$.
- **CBM_index** (*int*) – highest index of the integrated_dos array where the energy is less than *CBM*.

Returns the energy (eV, referenced from the CBM) to which the electrons will fill. For the case where the concentration specified would fill all states specified by integrated_dos, None is returned.

Return type float

Notes

The precision of the result will depend upon the energy resolution in the DOSCAR.

effmass.dos.**find_dos_CBM_index**(*Data*)

Finds the highest index of the integrated_dos array where the energy is less than *CBM*.

Parameters **Data** (Data) – Instance of the Data class.

Returns the highest index of the integrated_dos array where the energy is less than *CBM*.

Return type int

effmass.dos.**find_dos_VBM_index**(*Data*)

Finds the lowest index of the integrated_dos array where the energy exceeds *VBM*.

Parameters **Data** (Data) – Instance of the Data class.

Returns the lowest index of the integrated_dos array where the energy exceeds *VBM*.

Return type int

effmass.dos.**hole_fill_level**(*Data*, *volume*, *concentration*, *VBM_index*)

Finds the energy to which a given hole concentration will fill the density of states in integrated_dos.

Uses linear interpolation to estimate the energy between two points given in the DOSCAR.

Parameters

- **Data** (Data) – Instance of the Data class.
- **volume** (*float*) – volume of the unit cell in angstrom ^{3}.
- **concentration** – hole concentration in cm $^{-3}$.
- **VBM_index** (*int*) – lowest index of the integrated_dos array where the energy is more than than *VBM*.

Returns the energy (eV, referenced from the VBM) to which the holes will fill. For the case where the concentration specified would fill all states specified by integrated_dos, None is returned.

Return type float

Notes

The precision of the result will depend upon the energy resolution in the DOSCAR.

A module for analysing DOSCAR data.

2.4.4 effmass.analysis

A module for analysing the data contained in a *Segment* object.

Contains the *Segment* class and methods for calculating various definitions of the effective mass.

class effmass.analysis.**Segment**(*Data*, *band*, *kpoint_indices*)

Class for segments of the bandstructure. A Segment contains data for a particular region of reciprocal space and particular band.

band

The band number of the segment (counting starts from 0).

Type int

kpoint_indices

The indices of *effmass.inputs.Data.kpoints* from which this Segment is formed.

Type list(int)

kpoints

2-dimensional array. Each row contains the fractional coordinates of a kpoint [kx,ky,kz]. A slice of *effmass.inputs.Data.kpoints*.

Type array(float)

cartesian_kpoints

2-dimensional array. Each row contains the cartesian coordinates (angstrom $^{-1}$) of a kpoint.

Type array(float)

dk_angs

1-dimensional array which contains the distance (angstrom $^{-1}$) between each kpoint and the extrema.

Type array(float)

dk_bohr

1-dimensional array which contains the distance (bohr $^{-1}$) between each kpoint and the extrema.

Type array(float)

energies

1-dimensional array which contains the energy (eV) of each eigenstate. A slice of *effmass.inputs.Data.energies*.

Type array(float)

dE_eV

1-dimensional array which contains the difference in energy (hartree) between each kpoint and the extrema.

Type array(float)

dE_hartree

1-dimensional array which contains the difference in energy (eV) between each kpoint and the extrema.

Type array(float)

occupancy
2-dimensional array. Each row contains occupation number of the eigenstates for a particular band. A slice of *effmass.inputs.Data.occupancy*.

Type array(float)

direction
1-dimensional array with length 3. The direction between kpoints in the segment.

Type array(float)

band_type
The band type, determined by occupancy of the eigenstate. Argument choices are "conduction_band", "valence_band" or "unknown". If set to "unknown", some class methods will raise a warning and return None.

Type str

fermi_energy
the fermi energy in eV.

Type float

__init__ (*Data*, *band*, *kpoint_indices*)
Initialise an instance of the Segment class.

Parameters

- **Data** (*Data*) – Data instance initialised from the bandstructure which contains the segment
- **band** (*int*) – the band number of the segment
- **kpoint_indices** (*list(int)*) – the kpoint indices of the segment

Returns None.

alpha (*polyfit_order=6*, *truncate=True*)
The transport mass ($\frac{k}{\delta E \delta k}$) is calculated as a function of *dk_bohr* and fitted to a straight line. The gradient of this line determines the alpha parameter which is used in the kane dispersion.

See *effmass.analysis.Segment.transport_effmass()*.

Parameters

- **polyfit_order** (*int, optional*) – order of polynomial used to approximate the dispersion. Defaults to 6.
- **truncate** (*bool, optional*) – If True, data only up to *effmass.analysis.Segment.explosion_index()* is used. If False, alpha is calculated using data for the whole segment. Defaults to True.

Returns The alpha parameter (hartree $^{-1}$).

Return type float

explosion_index (*polyfit_order=6*)
This will find the index at which there is a change in sign of the second derivative.

In the region of this point the first derivative will pass through zero and so the transport mass ($\frac{1}{\delta E \delta k}$) will explode.

Parameters **polyfit_order** (*int, optional*) – order of polynomial used to approximate the dispersion. Defaults to 6.

Notes

This marks the point at which the Kane dispersion is definitely not valid, although it may be that the Kane dispersion is a poor approximation prior to this.

fd(*k*, *fermi_level*, *temp*, *alpha*, *mass_bandedge*)
Calculates the probability that an eigenstate of momentum k is occupied, using Fermi-Dirac statistics and assuming a Kane dispersion.

Parameters

- **k** – the momentum of the eigenstate (bohr:math:^{-1}).
- **fermi_level** (*float, optional*) – Fermi level (eV) to be used in Fermi-dirac statistics. Defaults to *fermi_energy*.
- **temp** (*float, optional*) – The temperature (K) to be used in Fermi-Dirac statistics. Defaults to 300.
- **alpha** (*float, optional*) – The alpha parameter of the Kane dispersion (hartree^{-1})
- **mass_bandedge** – The mass at bandedge parameter of the Kane dispersion (units electron mass).

Returns The probability that the eigenstate is occupied.

Return type float

Note: The sign of the alpha parameter and mass_bandedge are important. If these are negative (as would be expected for the valence band), then they must be passed as negative values to the function.

---

finite_difference_effmass()
The curvature at the band edge is calculated using a second order forward finite difference method. This is then inverted to give an effective mass.

Parameters **None** –

Returns Bandedge effective mass from finite difference (in units of electron mass).

Return type float

finite_difference_fit()
Calculates the curvature at the band edge using a finite difference method and then evaluates the corresponding quadratic dispersion along *dk_bohr*.

See *effmass.analysis.Segment.finite_difference_effmass()*.

Parameters **None** –

Returns list containing energies (hartree). The energies are calculated at 100 points evenly distributed across *dk_bohr* using the quadratic approximation.

Return type list(float)

five_point_leastsq_effmass()
Fits a parabolic dispersion using the least-squares method to 5 points (3 DFT-calculated points + 2 from symmetry).

Parameters **None** –

Returns Curvature effective mass (in units of electron mass).

Return type float

Notes

no weighting is used.

five_point_leastsq_fit()
Calculates the curvature effective mass using a parabolic least- squares fit and then evaluates the corresponding parabolic dispersion along *dk_bohr*.

Parameters **None** –

Returns list containing energies (hartree). The energies are calculated at 100 points evenly distributed across *dk_bohr*.

Return type list(float)

inertial_effmass(*polyfit_order=6*, *dk=None*, *polyfit_weighting=False*)
Calculates the inertial (curvature) effective mass ($\frac{1}{\frac{\delta^2 E}{\delta k^2}}$), evaluated at *dk_bohr*.

Parameters

- **polyfit_order** (*int, optional*) – order of polynomial used to approximate the dispersion. Defaults to 6.
- **dk** (*array, optional*) – distance (bohr :math:^{-1}) from extrema in reciprocal space at which to evaluate second order derivatives. Defaults to 100 points evenly distributed across the whole segment.
- **polyfit_weighting** (*bool, optional*) – If True, polyfit will be weighted according to occupation of eigenstates. If False, no weighting will be used.

Returns array(float). 1d array containing the conductivity effective mass (in units of electron rest mass) evaluated at the points specified in dk.

kane_fit(*polyfit_order=6*)
Calculate the Kane quasi-linear dispersion parameters, then evaluates at 100 points evenly distributed across *dk_bohr*.

The Kane quasi-linear dispersion is described by

..math:: frac{hbar ^2 k^2}{2m_{tb}} = E(1+alpha E)

where the transport mass at bandedge (m_{tb}) and the alpha parameter are calculated by fitting a linear function to th

..math:: m_t = m_{tb}(1+2alpha E)

The transport mass m_t is calculated by approximating the dispersion with a polynomial function and taking the first derivative, see *transport_effmass()*.

Parameters **polyfit_order** (*int, optional*) – order of polynomial used to approximate the dispersion. Defaults to 6.

Returns 1d array containing energies (hartree).

Return type array

kane_mass_band_edge(*polyfit_order=6*, *truncate=True*)
The transport mass ($\frac{1}{\delta E \delta k}$) is calculated as a function of *dk_bohr* and fitted to a straight line. The intercept of this line with the y-axis gives a transport mass at bandedge which is used as a parameter in the kane dispersion.

See *effmass.analysis.Segment.transport_effmass()*.

Parameters

- **polyfit_order** (*int, optional*) – order of polynomial used to approximate the dispersion. Defaults to 6.
- **truncate** (*bool, optional*) – If True, data only up to *effmass.analysis.Segment.explosion_index()* is used. If False, alpha is calculated using data for the whole segment. Defaults to True.

Returns transport mass at bandedge (in units of electron mass).

Return type float

mass_integration(*fermi_level=None*, *temp=300*, *alpha=None*, *mass_bandedge=None*, *upper_limit=None*)

Integrates the product of the fermi-dirac distribution, density of states and second derivative of kane dispersion along the one- dimensional slice of k-space defined by *Segment* (up to *explosion_index()*).

Parameters

- **fermi_level** (*float, optional*) – Fermi level (eV) to be used in Fermi-dirac statistics. Defaults to *fermi_energy*.
- **temp** (*float, optional*) – The temperature (K) to be used in Fermi-Dirac statistics. Defaults to 300.
- **alpha** (*float, optional*) – The alpha parameter of the Kane dispersion (hartree^{-1})
- **mass_bandedge** – The mass at bandedge parameter of the Kane dispersion (units electron mass).
- **upper_limit** (*float, optional*) – The integration upper limit (bohr^{-1}). Defaults to where the Kane quasi-linear dispersion is no longer valide, defined by *explosion_index()*.

Returns The optical effective mass (units of electron mass), defined as the inverse of the second derivative of a kane dispersion, weighted according to occupancy of available eigenstates (the product of density of states and the fermi-dirac distribution).

Return type float

Note: The sign of the alpha parameter and mass_bandedge are important. If these are negative (as would be expected for the valence band), then they must be passed as negative values to the function.

---

optical_effmass_kane_dispersion(*fermi_level=None*, *temp=300*, *alpha=None*, *mass_bandedge=None*, *upper_limit=None*)

Calculates the optical effective mass, with the dispersion approximated by a Kane quasi-linear function.

This optical effective mass is defined as:

..math:: frac{1}{m_o} = frac{int f(E_k(k),T) frac{delta^2 E_k(k)}{delta k^2} dk}{int f(E_k(k),T) dk}

where the integral is along the one-dimensional slice of k-space defined by *Segment* (up to *explosion_index()*)

..math:: frac{hbar ^2}{2m_{tb}} = E(1+alpha E)

where the transport mass at bandedge (m_{tb}) is calculated using *effmass.analysis.Segment.kane_mass_band_edge()* and the alpha parameter is calculated using *effmass.analysis.Segment.alpha()*.

Parameters

- **fermi_level** (*float, optional*) – Fermi level (eV) to be used in Fermi-dirac statistics. Defaults to *fermi_energy*.
- **temp** (*float, optional*) – The temperature (K) to be used in Fermi-Dirac statistics. Defaults to 300.
- **alpha** (*float, optional*) – The alpha parameter of the Kane dispersion (hartree^{-1})
- **mass_bandedge** – The mass at bandedge parameter of the Kane dispersion (units electron mass).
- **upper_limit** (*float, optional*) – The integration upper limit (bohr^{-1}). Defaults to where the Kane quasi-linear dispersion is no longer valide, defined by *explosion_index()*.

Returns The optical effective mass (units of electron mass) of the *Segment*.

Return type float

Note: The sign of the alpha parameter and mass_bandedge are important. If these are negative (as would be expected for the valence band), then they must be passed as negative values to the function.

---

optical_poly_effmass(*polyfit_order=6*)

Calculates the optical effective mass with a polynomial approximation to the dispersion

This optical effective mass is defined as:

..math:: frac{1}{m_o} = frac{sum_i f(E_i) g(k_i) frac{delta^2 E}{delta k^2}|_i}{sum f(E_i) g(k_i)}

where the sum is over eigenstates i contained withing the *Segment*. $f_(E_i)$ is the probability of occupation (Fermi-Dirac statistics) and $g(k_i)$ is the density of states at that k-point.

Parameters **polyfit_order** (*int, optional*) – order of polynomial used to approximate the dispersion. Defaults to 6.

Returns The optical effective mass (units of electron mass) of the *Segment*.

Return type float

poly_derivatives(*polyfit_order=6*, *polyfit_weighting=True*, *dk=None*)

Constructs a polynomial function using a least squares fit to Segment dispersion data, then evaluates first and second order derivatives.

Parameters

- **polyfit_order** (*int, optional*) – order of polynomial used to approximate the dispersion. Defaults to 6.
- **polyfit_weighting** (*bool, optional*) – If True, polyfit will be weighted according to occupation of eigenstates. If False, no weighting will be used.
- **dk** (*array, optional*) – distance (bohr :math:^{-1}) from extrema in reciprocal space at which to evaluate first and second order derivatives. Defaults to 100 points evenly distributed across the whole segment.

Returns A tuple containing a 1d array of first derivatives and 1d array of second derivatives, evaluated at dk: ([dedk],[d2edk2])

Return type tuple

poly_fit(*polyfit_order=6*, *polyfit_weighting=True*)

Constructs a polynomial function using a least squares fit to *Segment* dispersion data, then evaluates at 100 points evenly distributed across *dk_bohr*.

Parameters

- **polyfit_order** (*int, optional*) – order of polynomial used to approximate the dispersion. Defaults to 6.
- **polyfit_weighting** (*bool, optional*) – If True, polyfit will be weighted according to occupation of eigenstates. If False, no weighting will be used.

Returns 1d array containing energies (hartree).

Return type array

transport_effmass(*polyfit_order=6*, *dk=None*, *polyfit_weighting=False*)

Calculates the transport mass ($\frac{k}{\delta E \delta k}$), evaluated at *dk_bohr* .

Parameters

- **polyfit_order** (*int, optional*) – order of polynomial used to approximate the dispersion. Defaults to 6.
- **dk** (*array, optional*) – distance (bohr :math:^{-1}) from extrema in reciprocal space at which to evaluate first order derivatives. Defaults to 100 points evenly distributed across the whole segment.
- **polyfit_weighting** (*bool, optional*) – If True, polyfit will be weighted according to occupation of eigenstates. If False, no weighting will be used.

Returns array(float). 1d array containing the transport effective mass (in units of electron rest mass) evaluated at the points specified in dk.

weight_integration(*fermi_level=None*, *temp=300*, *alpha=None*, *mass_bandedge=None*, *upper_limit=None*)

Integrates the product of the fermi-dirac distribution along the one-dimensional slice of k-space defined by *Segment* (up to *explosion_index()*).

Parameters

- **fermi_level** (*float, optional*) – Fermi level (eV) to be used in Fermi-dirac statistics. Defaults to *fermi_energy*.
- **temp** (*float, optional*) – The temperature (K) to be used in Fermi-Dirac statistics. Defaults to 300.
- **upper_limit** (*float, optional*) – The integration upper limit (bohr^{-1}). Defaults to where the Kane quasi-linear dispersion is no longer valide, defined by *explosion_index()*.

Returns A normalisation factor for *mass_integration()*.

Return type float

Note: The sign of the alpha parameter and mass_bandedge are important. If these are negative (as would be expected for the valence band), then they must be passed as negative values to the function.

---

weighted_leastsq_effmass()
Fits a parabolic dispersion using the weighted least-squares method to all points in the segment, plus those from symmetry, E(k)=E(-k).

Parameters **None** –

Returns Curvature effective mass (in units of electron mass)

Return type float

Notes

weighting is given by the Fermi-Dirac distribution.

weighted_leastsq_fit()
Calculates the curvature effective mass usinhag a weighted least- squares fit and then evaluates the corresponding parabolic dispersion along *dk_bohr*.

Parameters **None** –

Returns list containing energies (hartree). The energies are calculated at 100 points evenly distributed across *dk_bohr*.

Return type list(float)

weighting(*fermi_level=None*, *temp=300*)
Calculates a weighting for each kpoint using the Fermi-Dirac statistics.

Parameters

- **quasi_fermi_level** (*float, optional*) – The fermi level to be used in Fermi-Dirac statistics. Defaults to `effmass.inputs.Segment.fermi_energy`.
- **temp** (*float, optional*) – The temperature (K) to be used in Fermi-Dirac statistics. Defaults to 300.

Returns A 1-dimensional array which contains the weighting for each k-point.

Return type array(float)

Notes

The weighting is relative only: constants will cancel out when using to weight least square fits or means.

class effmass.analysis.**SegmentVasp**(*DataVasp*, *band*, *kpoint_indices*)
Class for segments of a Vasp bandstructure. Inherits from *Segment* class, and extends to include DOS Segment data.

Additional attributes: dos (array): 2-dimensional array. Each row contains density of states data (units "number of states / unit cell") at a given energy: [energy(float),dos(float)]. A slice of `effmass.inputs.DataVasp.dos`. integrated_dos (array): 2-dimensional array. Each row contains integrated density of states data at a given energy: [energy(float),integrated_dos(float)]. A slice of `effmass.inputs.DataVasp.integrated_dos`.

__init__(*DataVasp*, *band*, *kpoint_indices*)
Initialise an instance of the Segment class.

Parameters

- **Data** (Data) – Data instance initialised from the bandstructure which contains the segment

- **band** (*int*) – the band number of the segment
- **kpoint_indices** (*list (int)*) – the kpoint indices of the segment

Returns None.

A module for analysing the data contained in a *Segment* object.

Contains the *Segment* class and methods for calculating various definitions of the effective mass.

2.4.5 effmass.outputs

A module for plotting and summarising segments information, density-of-states information and effective mass analysis.

effmass.outputs.**make_table**(*segments*, *which_values=None*)

Prints table summary of segments data to terminal

Parameters

- **segments** (*list*) – Which segments to use.
- **which_values** (*list*) – use 'least squares' or 'finite differences' (default: both)

effmass.outputs.**plot_dos**(*DataVasp*, *figsize=(8, 8)*)

Plots density of states (states/unit-cell) against energy (eV).

Parameters

- **DataVasp** (DataVasp) – instance of the DataVasp class.
- **figsize** (*list*) – Size of matplotlib figure. Default: (8, 8)

Returns tuple containing instance of the matplotlib.pyplot.figure class and matplotlib.pyplot.axes class.

Return type Figure, Axes

Notes

The valence band maximum is set to 0 eV.

effmass.outputs.**plot_integrated_dos**(*DataVasp*, *figsize=(8, 8)*)

Plots integrated density of states (states/unit-cell) against energy (eV).

Parameters

- **DataVasp** (DataVasp) – instance of the DataVasp class.
- **figsize** (*list*) – Size of matplotlib figure. Default: (8, 8)

Returns

tuple containing instance of the matplotlib.pyplot.figure class and matplotlib.pyplot.axes class.

Return type Figure, Axes

Notes

The valence band maximum is set to 0 eV.

effmass.outputs.**plot_segments**(*Data*, *Settings*, *segments*, *savefig=False*, *random_int=None*, *figsize=(8, 8)*)

Plots bandstructure overlaid with the DFT-calculated points for each Segment instance. Each Segment is labelled with it's direction in reciprocal space and index number from the segments argument.

Parameters

- **Data** (Data) – instance of the Data class.
- **Settings** (Settings) – instance of the Settings class.
- **segments** (*list (Segment)*) – A list of instances of the Segment class.
- **figsize** (*list*) – Size of matplotlib figure. Default: (8, 8)

Returns

tuple containing instance of the matplotlib.pyplot.figure class and matplotlib.pyplot.axes class.

Return type Figure, Axes

Notes

The x-axis of the plot is not to scale.

A module for plotting and summarising segments information, density-of-states information and effective mass analysis.

2.5 Implementation

2.5.1 Methods for calculating the curvature effective mass

Three methods are used to calculate the curvature effective mass (Eqn.1 in the main text).

- **Finite difference**

  We use a three point forward finite difference equation to calculate the curvature at point i:

  $$\frac{\partial^2 E}{\partial k^2} = \frac{E_{i+2} - 2E_{i+1} + E_i}{|k_{i+1} - k_i|},$$

  where E_i is the energy eigenvalue at position k_i in reciprocal space.

- **Unweighted least-squares fit**

  To obtain estimates for the coefficient of a parabolic dispersion

  $$E = ck^2,$$

  we use the least-squares method as implemented in the NumPy Python library to minimise the summed square of residuals

  $$\sum_{i=1}^{5} (ck_i^2 - E_i)^2.$$

  We fit to five points; three points from the DFT-calculated dispersion plus two from the symmetry of the dispersion ($E(k) = E(-k)$).

- **Weighted least-squares fit**

  To obtain estimates for the coefficients of the dispersion

  $$E = ck^2,$$

  we use the least-squares method as implemented in the NumPy Python library to minimise the summed square of residuals

  $$\sum_{i=1}^{n} W_i(ck_i^2 - E_i)^2.$$

  The summation is over all points up to an energy of $0.25\,eV$, including points generated from the symmetry of the dispersion, $E(k) = E(-k)$. W_i is given by

  $$W_i(E_i, T) = \frac{1}{\exp\left(\frac{E_i - E_f}{k_B T}\right) + 1}.$$

2.6 Related packages

The *effmass* package is aimed towards theoretical solid state physicists and chemists who have a basic familiarity with Python. Depending on the functionality and level of approximation you are looking for, it may be that one of the packages listed below will suit your needs better.

vasppy: This is installed as a dependancy of *effmass*. Calculates the effective mass using a least-squares quadratic fit for parabolic dispersions.

sumo: Calculates the effective mass using a least-squares fit for parabolic and non-parabolic dispersions.

emc: Calculates the effective mass *tensor* using a finite-difference method for parabolic dispersions.

pymatgen: This is installed as a dependancy of *effmass*. Calculates an average effective mass *tensor* for non-parabolic dispersions with multiple bands and extrema. Also calculates the Seebeck effective mass as defined here.

mstar: Effective mass calculator using k.p perturbation theory. Assumes parabolicity; provides the conductivity effective mass tensor at band edge. The advantage of this code is that it does not require a DFT calculation along the high symmetry paths in reciprocal space.

EMAF: Calculates the DOS effective mass (at a specific temperature), taking into account non-parabolicity and anisotropy. Also calculates the conductivity effective mass. Written in Matlab.

If you have an update to the information above then please use the Github issue tracker.

2.6.1 Which features are unique to the *effmass* package?

To our knowledge, the following features are unique to *effmass*:

- easily compare the values of curvature effective mass calculated using multiple numerical techniques (least-squares and polynomial fitting)
- tailor the polynomial fitting used to approximate the DFT calculated dispersion: by choosing the order of the polynomial and the energy range to fit over.
- visualise the dispersions used to approximate the DFT calculated dispersion
- quantify non-parabolicity through the Kane dispersion parameters: effective mass at band-edge and alpha
- calculate the optical effective mass assuming a Kane dispersion.

2.7 Development

Please use the Github issue tracker for feature requests and bug reports.

If you would like to contribute please do so via a pull request. All contributors must read and respect the code of conduct. In particular, we welcome contributions which would extend *effmass* so that it is able to parse output from other electronic structure codes.

2.8 Testing

Automated testing of the latest commit happens here.

Manual tests can be run using

```
python3 -m pytest
```

This code has been tested with Python version 3.6.

2.9 Citing

If you use this code in your research, please cite the following paper:

Whalley, Lucy D. (2018). *effmass - an effective mass package*. The Journal of Open Source Software, 3(28) 797.

Citation `bibtex`

2.10 License

2.11 Acknowledgements

This package was originally written by Lucy Whalley during a PhD funded by the EPSRC through the Centre for Doctoral Training in New and Sustainable Photovoltaics (grant no. EP/L01551X/1) at Imperial College London. Aron Walsh and Benjamin J. Morgan supervised the development of the code.

Python Module Index

e

Index

Symbols

A

B

C

D

E

F

G

H

I

K

M

N

O

P

R

S

T

V

W