Effective Mass Calculator

Effective mass calculator (**EMC**) implements calculation of the effective masses at the bands extrema using finite difference method (**not** the band fitting method). There are currently two versions of the script: FORTRAN program and Python script. Currently *CRYSTAL* and *VASP* are supported, *Quantum Espresso* is coming!

Continuous integration

build passing

Theory

Effective mass (m*) is defined as:

$$\left(\frac{1}{m^*}\right)_{ij} = \frac{1}{\hbar^2}\frac{\partial^2 E_n(\vec{k})}{\partial k_i k_j},\ i,j = x,y,z$$

where *x, y, z* are the directions in the reciprocal Cartesian space (2π/A), *En(k)* is the dispersion relation for the *n*-th electronic band. The explicit form of the right-side symmetric tensor from the above eq. is:

$$\frac{d^2E}{dk^2} = \begin{pmatrix} \frac{d^2E}{dk_x^2} & \frac{d^2E}{dk_x dk_y} & \frac{d^2E}{dk_x dk_z} \\ . & \frac{d^2E}{dk_y^2} & \frac{d^2E}{dk_y dk_z} \\ . & . & \frac{d^2E}{dk_z^2} \end{pmatrix}$$

where the derivatives can be evaluated numerically using the finite difference method. Eigenvalues of the above matrix are inverses of the effective masses, eigenvectors are the directions of the principal effective mass components.

For the theory behind the code and validation of the code against known data see the paper. Let us know if you find any bugs or mistakes, thanks!

Notes

1. Atomic units (a.u.) are used throughout the code: ħ = 1, energy is in Hartree, effective mass is in the electron mass at rest (m0), distance is in Bohr.
2. For the top of the VB (valence band) eigenvalues are negative, for the bottom of the CB (conduction band) eigenvalues are positive.
3. In some cases, not all eigenvalues have the same sign, meaning that the chosen k-point is not a global minimum (maximum).
4. Effective masses can be highly anisotropic (see Running tests section).

Installation

EMC is a Python script, that depends only on the Python Standard Library.
To install:

- download the latest version
- unpack it: `tar -zxvf emc.tar.gz`
- check that *emc.py* has executable flag using `ls -la`, if it doesn't do `chmod +x ./emc.py`

- check that *emc.py* is in your path `$PATH` by printing the `$PATH` variable: `echo $PATH`
- enjoy the results!

Requirements

The code is being tested with Python v 2.7 (as you can see in the travis-ci config file). Other versions of the Python will be coming soon.

Input file structure

```
0.000 0.000 0.000                             ! K-POINT in the reciprocal crystal coord. (3 floats)
0.01                                          ! step size in 1/Bohr units (1 float)
81                                            ! band number, (1 integer)
V                                             ! program identifier (1 char)
6.291999817  0.000000000  0.000000000         ! direct lattice vectors (3 floats)
0.755765092  7.652872670  0.000000000         ! direct lattice vectors (3 floats)
0.462692761  3.245907103 14.032346772         ! direct lattice vectors (3 floats)
```

- **band number**. If *CRYSTAL* is employed, band number should be set to **1**. Helper script `cry-getE.pl` reads-in the desired band number (see below). For *VASP* valence band number can be obtained as a half of the `NELECT` variable from the *OUTCAR* file (for non spin-polarized calculations).
- **program identifier**: `C` for *CRYSTAL* or `V` for *VASP*. (TODO: Quantum Espresso)
- **direct lattice components** in *CRYSTAL* can be found in the SCF output under: `DIRECT LATTICE VECTORS COMPON. (A.U.)`, in *VASP* in the *OUTCAR* under: `direct lattice vectors`.

Usage

1. Run SCF.
2. Generate k-point grid (in *KPOINT* file) using *emc.py* with the appropriate input file.
3. Run non-self consistent calculation using obtained grid:
    - in case of CRYSTAL: run helper script cry-getE.pl (see below)
    - in case of VASP: set `ICHARG=11` in the *INCAR*. Don't forget to copy *CHGCAR* file from the converged SCF run.

Helper script cry-getE.pl

In case of *CRYSTAL*, *cry-getE.pl* script should be used in order to obtain file with the energies on the grid. The script takes two k-points at a time and runs band structure calculations (using *runprop* script from the *CRYSTAL* package).

cry-getE.pl has the following command line options:

- `-f` SCF output filename (.f9)
- `-b` band number

Example: `cry-getE.pl -f ../input.f9 -b 131`

Note that `runprop` needs to be in the current `$PATH`, otherwise script will quit.

Running tests

To run tests, in the distribution directory run:
```
python -m unittest discover
```

Tests are located in the *test* folder.

Authors

Alexandr Fonari and Christopher Sutton

License: MIT