2D Schroedinger-Poisson Solver: Documentation

July 6, 2016

1 Introduction

This code aims at self-consistently solving the coupled Schroedinger-Poisson equations in 2D materials. More accurately, it is designed to simulate nanosheets of a single material with regions of different strains. The Schroedinger equation is in 1D, meaning that the strain can vary on a single axis (x), as illustrated in figure 1. Moreover, the material is assumed to be infinite in the y-direction and periodic in the x-direction.

Two types of calculations can be made: single-point or map. The former precisely simulates a specific setup described in an input file to produce a band or a carrier density profile. This is a useful tool for experiments as it allows to simulated any strain profile (almost continuously). The map calculation screens over different setups of strained/unstrained material to investigate insulator to conductor phase transitions.

This code uses a multi-scale approach, meaning that the materials properties must come from DFT calculations. The user has to provide vital quantities such as effective masses and polarization charges for different strains. The properties for at least 4-5 strains should be provided for accurate fitting. It is also not recommended to simulate strains larger than those coming from DFT, as the trends could change.

2 Setting up the code

The code itself is contained in 3 files: `2Dschrpoisson.py`, `schrpoisson_wire2.f90` and `Makefile`. The bulk of the numerical part being written in FORTRAN, some steps must be taken before using the code. Firstly, the Python-Fortran interface must be created using `f2py` in the code folder:

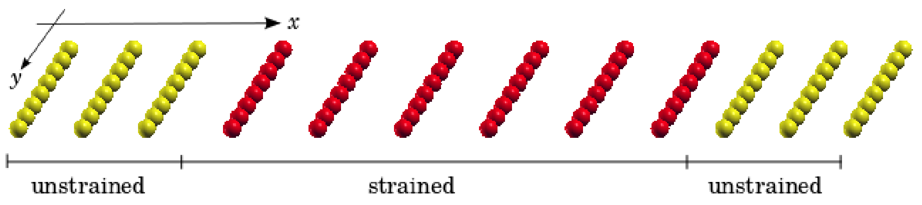

Figure 1: Example of simulation setup: Periodic alternation of strained and unstrained material

```
f2py -m schrpoisson_wire2 -h schrpoisson_wire2.pyf schrpoisson_wire2.f90
```

One must then compile using the `make` command. The code is now ready to run using the following command in the code folder:

```
python 2Dschrpoisson.py {material_properties}.json {calculation_input}.json
```

3 Input

Two input files are necessary to evry calculation: one containing the materials properties and the other the calculation setup. All input files must be stored in the input folder.

3.1 Materials Properties File

The file containing the properties of the material at different strains must be in the `json` format, in the form of a python dictionnary. The following example is taken from the SnSe_example.json in the input folder. Note that the properties of the unstrained material are **mendatory** and must be given under the "0.00" key.

```
{"0.00": { "alpha_xx": 10.22,
           "x_lat": 4.408,
           "y_lat": 4.288,
           "polarization_charge": 0.0,
           "vacuum_level": 3.471695,
           "valence_gamma": { "energy": -1.508255819,
                              "conf_mass": 1.755925079,
                              "DOS_mass": 2.7330924,
                              "degeneracy": 1
                              },
           "valence_gamma-X": {"energy": -0.8832657543,
                               "conf_mass": 0.125168454,
                               "DOS_mass": 0.159070572,
                               "degeneracy": 2
                               },
           "valence_gamma-Y": {"energy": -1.064317364,
                               "conf_mass": 0.109554071,
                               "DOS_mass": 0.159511425,
                               "degeneracy": 2
                               },
           "conduction_gamma": {"energy": 0.6090962485,
                                "conf_mass": 2.741100107,
                                "DOS_mass": 2.994158008,
                                "degeneracy": 1
                                },
           "conduction_gamma-X": {"energy": 0.0902128713,
                                  "conf_mass": 0.110807373,
                                  "DOS_mass": 0.190387132,
                                  "degeneracy": 2
                                  },
           "conduction_gamma-Y": {"energy": 0.05720314215,
                                  "conf_mass": 0.131542031,
                                  "DOS_mass": 0.130378261,
                                  "degeneracy": 2
                                  }
           },
"0.01": {...
         },
...}
```

- The "0.00" key: the first key is the strain of a the material and the assigned value is a subdictionary with all the relevant peoperties. The "0.00" key is mendatory.
- The "alpha_xx" key: this refers to the polarizability of the strained material in Å. x is the direction along which different strains/materials alternate
- The "x_lat" and "y_lat" keys: the lattice parameters of the unit cell in Å. Only needed for the unstrained material
- The "polarization_charge" key: the polarization charge at the edge of the material in units of electron per unit cell width, *i.e.* e/b.
- The "vacuum_level" key: Self-explanatory, need to align energy levels across the setup (in eV).
- The energy levels keys: They need to either contain the word `valence` or `conduction`, which is needed by the code. The assigned value is a subdictionary with the energy of the band extremun (eV), the confinement and density of states (DOS) masses (respectively effective masses in x and y direction) and the degeneracy of the energy level in the first BZ..

3.2 Calculation Input File

Each type of calculation , single-point or map, has its own input file template. However, both of them are required to be `json` files in the form of a python dictionary. The following examples can be found in the input folder and can be generated using the `create_calc_input.py` script.

3.2.1 Single-Point Calculation

```
{
  "calculation": "single_point",
  "out_dir": "single_point_output",
  "smearing": true,
  "KbT": 0.005
  "max_iteration": 1000,
  "plot_fit": false,
  "nb_of_states_per_band": 2,
  "delta_x": 0.5,
  "setup": {
            "slab1": {
                      "polarization": "positive",
                    "strain": 0.0,
                    "width": 15.0
            },
            "slab2": {
                    "polarization": "positive",
                    "strain": 0.08,
                    "width": 30.0
            },
            "slab3": {
                    "polarization": "positive",
                    "strain": 0.0,
                    "width": 15.0
            },
    },
}
```

- The "calculation" key: the value `"single_point"` indicates the type of calculation
- The "out_dir" key: indicates the name of the folder where the output data is dumped
- The "smearing" key: If `true`, a Marzari-Vanderbilt smearing is imposed for the electron distribution. It helps for the convergence of the self-consistent Schroedinger-Poisson loop and allows better comparisons with DFT calculations.
- The "KbT" key: In case of smearing, this is the imposed electronic temperature (Ry)
- The "max_iteration" key: The number of allowed steps for the self-consistent Schroedinger-Poisson loop, 1000 is a safe number.
- The "plot_fit" key: If `true`, the fitting of materials properties for any strain is displayed in a series of graphs
- The "nb_of_states_per_band" key: To limit the size of the output data. Here, only two quantum states per band (6 in total) will be reported
- The "delta_x" key: The size of the discretization step in space (Å)
- The "setup" key: The assigned value is a subdictionary with the description of each slab forming the general setup. For each layer, one has to specify the "polarization" (positive or negative, positive if holes accumulate towards larger x), the "strain" of the material and the "width" of the slab (Å). There is no limit to the numbers of layers but the keys must follow the logic, *i.e.* "slab+integer". It is not recommended to include strains higher than the largest one in the material properties file. It is also **mendatory** that the first and the last slabs have identical strain, since the code does not cope with polarization charges at the edges of the cell.

3.2.2 Map Calculation

```
{
 "calculation": "map",
 "out_dir": "map_output",
 "smearing": true,
 "KbT": 0.005,
 "max_iteration": 1000,
 "plot_fit": false,
 "nb_of_steps": 200,
 "upper_delta_x_limit": 0.5,
 "strain": {
            "max_strain": 0.1,
            "min_strain": 0.0,
            "strain_step": 0.01
 },
 "width": {
            "min_width": 10.0,
            "width_step": 5.0,
            "max_width": 60.0
 },
}
```

- The "calculation" key: the value `"single_point"` indicates the type of calculation
- The "out_dir" key: indicates the name of the folder where the output data is dumped
- The "smearing" key: If `true`, a Marzari-Vanderbilt smearing is imposed for the electron distribution. It helps for the convergence of the self-consistent Schroedinger-Poisson loop and allows better comparisons with DFT calculations.
- The "KbT" key: In case of smearing, this is the imposed electronic temperature (Ry)
- The "max_iteration" key: The number of allowed steps for the self-consistent Schroedinger-Poisson loop, 1000 is a safe number.
- The "plot_fit" key: If `true`, the fitting of materials properties for any strain is displayed in a series of graphs
- The "nb_of_steps" key: Since a wide range of setup sizes is spanned in a map calculation, a common step size might be a problem. Ii is instead dynamically adapted as $\Delta x = \frac{L_{tot}}{\text{nb\_of\_steps}}$
- The "upper_delta_x_limit" key: In case a limit should be imposed on the step size: $\Delta x = min(\frac{L_{tot}}{nb\_of\_steps}, \text{upper\_delta\_x\_limit})$
- The "strain" key: The assigned value is a subdictionary with the lower and upper bound ("min_strain" and "max_strain") of the strains over which calculations will be done.
- The "width" key: The assigned value is a subdictionary with the lower and upper bound ("min_width" and "max_width") of the width over which calculations will be done. The width refers to the unstrained material, the size of the strained slab is computed via $L_{strained} = (1 + \epsilon)L_{unstrained}$. In this example, 110 single-point calculations will be conducted with every combination of strain $\epsilon = 0.0, 0.01, ..., 0.1$ and width $L = 10.0, 15.0, ..., 60.0$.

4 Output

4.1 Single-Point Calculation

In single-point calculations, 3 output files are generated. The first one (`general_info.txt`) contains general information such as the number of iterations needed, convergence parameters or total free carrier density. The second one, `band_data.txt`, contains all the necessary information to produce a band profile (Fermi energy, potential profile of each band, their respective quantum states, etc.) Finally, the last file (`density_profile.txt`) contains the free electrons and holes density as a function of position, in different units.

4.2 Map Calculation

Map calculations produce a single output file in which, among others, the total electron density is given for each strain/width combination in order to produce a map.