BoltzTraP2 tutorial

Learning Objectives

- Go through some of the main features of BoltzTraP2 with specific examples.
- Familiarise yourself with the BoltzTraP2 output.
- Use BoltzTraP2 built-in tools to plot the band structure and electronic transport properties.

Introduction:

The aim of this tutorial is to enable you to compute the electronic thermoelectric properties from first principles with the help of CASTEP and BoltzTraP2. In order to do so, one needs to calculate the density of states on a very dense grid.

You will need:

CASTEP example files, BoltzTraP2 (Python 3 library).

Example files:

Copy the example files
`wget https://github.com/ganphys/castep2boltz/blob/master/BoltzTraP2-tutorial/BoltzTraP2-tutorial.tar.gz`
Then untar and unzip it using:
```
gunzip BoltzTraP2-tutorial.tar.gz
tar -xvf BoltzTraP2-tutorial.tar
```

Running BoltzTraP2:

We need to calculate the density of states with CASTEP before performing a BoltzTraP2 calculation. Navigate to the example directory, investigate briefly the `.cell` and `.param` files and run CASTEP. The calculation should be done within 1-2 minutes on a single core.

Next we focus on the command-line interface of BoltzTraP2, the btp2 script. You can get general help regarding the usage of btp2 simply by running:

```
btp2 -h
```

The most important part of each btp2 invocation is the choice of subcommand, i.e., of an action to perform. The help command above will present you with a list of subcommands. Obtaining specific help for one of them is also easy; for instance,

```
btp2 interpolate -h
```

will print detailed usage help for the *interpolate* subcommand.

We are ready to run btp2. For that purpose open a terminal in the example directory, where the CASTEP calculation is located and type:

```
btp2 -vv interpolate -m 5 .
```

The -vv flag is completely optional and has no effect on the calculation. It tells btp2 to print lots of diagnostic information to the terminal. In contrast, `-m 5` is a critical parameter controlling how fine the interpolation is. In this case, we tell the program to try and sample three irreducible **k**-points for each one that is present in the input.

After a while, the program will finish and save the results to an interpolation.bt2 file, which is just an xz-compressed JSON document. Most regular users will only interact with `.bt2` files through the btp2 interface, and can safely treat them as black boxes.

Plotting the band structure:

We can take a look at the results of the interpolation by plotting the reconstructed bands along a particular path in reciprocal space:

```
btp2 plotbands interpolation.bt2 ["[0.0, 0.0, 0.0], [0.5, 0.5, 0.5], [0.5, 0.0, 0.5]"]
```

This plots the band structure along the Γ-L-X.

TASK 1: Now plot the band structure along the W-L-Γ-X-W-K path and increase the sampling between points to 80.

Hint: You might want to use the btp2 help command for plotbands to see how the sampling can be increased.

The coordinates of the high-symmetry points are given below:

```
0.5 0.25 0.75      ! W
0.5 0.5 0.5        ! L
0.0 0.0  0.0       ! Γ
0.5 0.0 0.5        ! X
```

0.5 0.25 0.75 ! W
0.375 0.375 0.75 ! K
Compare the band structure to literature results. Are there any valence or conduction bands missing? To check if that is an expected behaviour you can return to the diagnostic information from the interpolation run and count how many bands near the Fermi level are included by btp2 during the interpolation.

While this band structure plot is not as precise as an equivalent band structure obtained via a dedicated CASTEP calculation, it provides a quick way of visualising different paths in the Brillouin zone.

Computing electronic transport

One of the most popular use cases of BoltzTraP2 is to estimate the phenomenological electronic transport of the system by integrating over its band structure. To do so, we start from the results of the interpolation step and use the *integrate* subcommand:

```
btp2 -vv integrate interpolation.bt2 300
```

This will calculate the electronic transport at 300 K using the default uniform-relaxation-time model. The output is saved in two different formats:

- A new compressed JSON file with the same name as the input but with a `.btj` extension. This is more convenient for automatic processing as it can be loaded in Python as a dictionary.
- A set of text files closely following the formats defined by the original BoltzTraP, with the extensions `.condtens, .halltens,` and `.trace`. They can be visualized with any text editor or pager, loaded into any data processing code or plotted with almost any software on the market. The first line in each file is a header describing the meaning and the units of each column.

TASK 2: Explore the temperature range from 200 K to 800 K in intervals of 50 K. You might want to use `btp2 integrate -h` to see how to accomplish the task.

TIP: There is a second subcommand, *dope*, which can be used instead of *integrate* to obtain sets of Onsager coefficients. It is more limited in scope and only generates "old-style" output (typical for version 1 of BoltzTraP). In other words you need to use another program to plot the results. The main use case for *dope* is to explore particular carrier concentrations instead of scanning over a range of chemical potentials. For example:

```
btp2 -vv dope interpolation.bt2 300 1e20:1e21:1e20
```

will calculate the electronic transport at 300 K for p-type doping from 1e20 cm^{-3} to 1e21 cm^{-3} in intervals of 1e20 cm^{-3}. For holes you need to use a negative sign when defining the doping levels.

Plotting the results

The coefficients computed via the integrate subcommand in the previous step can be plotted with btp2 itself. For instance, to plot the xx and zz components of the Seebeck coefficient as a function of the chemical potential μ, we run:

```
btp2 plot -u -c '["xx", "zz"]' -s 1 interpolation.btj S
```

Be careful with the quotes, since the shell must not strip them off of the string at the space. Here, `-u` selects μ as the independent variable (the other option is `-T`, for the temperature), and `-s` tells btp2 to subsample the other variable (T) and plot one curve for each N values. In this case N=1 because we want to plot a curve for every temperature in our sampling set. The last parameter selects which coefficient should be plotted; for a list of valid options, see the output of `btp plot -h`. Finally, in the case of tensor quantities like the Seebeck coefficients we need to specify which components we are interested in through the argument to `-c`. Note that the special value **`"scalar"`** inside the square brackets would calculate and plot 1/3Trace(S).

TASK 3: Plot the trace of the Power Factor (PF) for our silicon example from 200 K to 700 K in increments of 100 K.

TIP: You might notice that there are not a lot of data points near the Fermi level. You can improve this by including the `-b` parameter (stands for bins) when you are computing the transport properties with integrate. You can try with `-b 1000`. Roughly speaking, having more bins will increase the number of data points but it will also increase the noise in your results. Therefore, if your **k**-points sampling is not dense enough, using more bins might actually give you worse results.

If the time allows you, repeat the CASTEP calculation with `kpoints_mp_grid 8 8 8` and `spectral_kpoints_mp_grid 40 40 40`. It should take you around 10 minutes on 4 cores. Now you can compare the two sets of results and see how having a denser **k**-points mesh improves the quality of the btp2 results.

Additional information:

This tutorial is based on the BoltzTraP2 wiki guide, which can be found here: https://gitlab.com/sousaw/BoltzTraP2/wikis/tutorial

There you can find more information about BoltzTraP2, see an example with LiZnSb and learn how to display the Fermi surface with btp2.