ShengBTE tutorial

Learning Objectives

- Gain awareness and familiarity with the tools available to compute lattice thermal conductivity with CASTEP.
- Learn how to prepare ShengBTE input files from data obtained with CASTEP.
- Familiarise yourself with the ShengBTE computational workflow and final output.

Introduction:

The aim of this tutorial is to enable you to compute the lattice thermal conductivity from first principles with the help of CASTEP and ShengBTE. In order to do so one needs to know the second- and third-order force constants of the system.

First you will investigate the CASTEP input files (`.cell` and `.param`) used for the phonon run which estimates the second-order force constants. Next you will convert the CASTEP data into a suitable format for ShengBTE. Then you will calculate the third-order force constants using the finite-displacement supercell approach. Finally, you will perform a ShengBTE calculation and plot the lattice thermal conductivity with respect to temperature.

You will need:

CASTEP example files, ShengBTE and thirdorder programs, castep2shengbte.py and thordorder_castep.py Python 2.7 scripts.

Example files:

Copy the example files

```
wget github.com/ganphys/castep2shengbte/blob/master/tutorial/ShengBTE-tutorial.tar.gz
```

Then untar and unzip it using:

```
gunzip ShengBTE-tutorial.tar.gz
tar -xvf ShengBTE-tutorial.tar
```

Running castep2shengbte.py:

You can run the CASTEP to ShengBTE interface by calling the script:

```
castep2shengbte.py <seedname> nx ny nz
```

where `nx ny nz` is the q-point grid. If executed successfully, the script will create 2 output files which can be used in ShengBTE: `CONTROL` and `FORCE_CONSTANTS_2ND.`

Task 1: Examine the `.cell` and `.castep` files in the "NbFeSb-2nd-IFC" directory and try to guess what q-point grid was used for the NbFeSb calculation. Try to locate the Force constants matrix block in the `.castep` file.

Now run the script with NbFeSb as the `<seedname>` and the correct q-point grid.

TIP: The parameter "`PHONON_WRITE_FORCE_CONSTANTS: true`" needs to be added to `.param` or the force constants will not be written in `.castep`. If you forget to add the parameter before the phonon calculation, simply include it to the `.param` file when the calculation is ready and make a continuation run. CASTEP will use the `.check` file to recover and display the force constants.

Running thirdorder with CASTEP:

Third-order force constants are also required by ShengBTE. These are generated by the thirdorder_castep.py program and are stored in a file called `FORCE_CONSTANTS_3RD`.

The calculation process is as follows:

1. thirdorder_castep.py creates a set of supercells containing atoms with perturbed positions, and writes CASTEP input files for each one.
2. CASTEP computes the forces between atoms for all of the supercells in separate runs, using an energy calculation.
3. thirdorder_castep.py gathers the forces from the CASTEP output of each run and constructs the third-order force constants.

IMPORTANT: The default energy tolerance for the electronic minimisation in the energy calculations must be reduced below the default value and set on par with the default value used in the phonon calculations. Otherwise, noise will dominate the third-order force constants.

The thirdorder_castep.py script takes exactly six mandatory command-line arguments:

```
thirdorder_castep.py sow|reap na nb nc cutoff[nm/-integer] <seedname>
```

The first argument must be either "sow" or "reap", and chooses the operation to be performed (“sow” is the displacement generation; “reap” is the irreducible force constant (IFC) matrix reconstruction). The next three must be positive integers, and specify the dimensions of the supercell to be created. The "cutoff" parameter specifies the force cutoff distance in nanometres; interactions between atoms further apart than this parameter are neglected. If cutoff is a negative integer -n, the cutoff is set automatically to the maximum distance of the n-th nearest neighbours in the supercell, e.g. if it is set to -3, the 3rd nearest-neighbour distances will be computed, and the cutoff set to the largest value. Finally, `<seedname>` is the file prefix for CASTEP's input/output files.

Any invocation of thirdorder_castep.py requires a CASTEP `.param` and `.cell` file with a description of the parameters and unit cell to be present in the current directory.

Now examine the `.param` and `.cell` files for our thirdorder calculation. They are located in the “NbFeSb-3rd-IFC” directory. Notice the change in the default energy tolerance for the electronic minimisation in the .param file. Why is that necessary?

To generate an irreducible set of displacements for a `2x2x2` supercell and up to third-nearest-neighbour interactions, we run:

```
thirdorder_castep.py sow 2 2 2 -3 NbFeSb
```

This creates a new “NbFeSb-3RD” directory (“`<seedname>`-3RD” in the general case) with the undisplaced supercell coordinates and 332 subdirectories with names following the pattern job-*, which contain supercells with small perturbations
to the atomic positions. Each job is a separate calculation which needs to be done with CASTEP.

TASK 2: Try to run thirdorder_castep.py in sow mode for different supercell sizes and different nearest-neighbour distances. Examine how that changes the number jobs in “`<seedname>`-3RD”. You might want to clean the content of “`<seedname>`-3RD” in between test runs.

TIP: Submitting so many jobs manually can be a very time consuming task. Therefore, it might be worth writing a bash script which would do that for you. As an example, on a given system the user could run the jobs in series with a command sequence like:

```
for i in {000..332}
do
 cd NbFeSb-3RD/job-$i
 aprun -n ${n} castep.mpi NbFeSb
 cd -
 echo "job-$i done" >> jobs_done.txt
done
```

It is necessary to complete all jobs in “`<seedname>`-3RD” directory before proceeding to the REAP step.

After the jobs have completed successfully, the output files have to be collated and passed to thirdorder_castep.py, this time in REAP mode. The general syntax is:

```
find <seedname>-3RD/job* -name <seedname>.castep | sort -n| thirdorder_castep.py reap nx ny nz cutoff <seedname>
```

For the purposes of this tutorial we will skip the computation part and use the precomputed results in the “NbFeSb-3RD-precomputed” directory. The REAP command in that case is:

```
find NbFeSb-3RD-precomputed/job* -name NbFeSb.castep | sort -n| thirdorder_castep.py reap 2 2 2 -3 NbFeSb
```

TIP: The interface prints a message in the terminal for every job which is read successfully. If there is something wrong the interface will terminate the process with an error. Please check the `.castep` file for the job which caused the error. It might be that the block with forces is missing or incomplete.

If everything goes well, a `FORCE_CONSTANTS_3RD` file will be created at the end of the run. Naturally, it is important to choose the same parameters (`nx, ny, nz, cutoff`) for the sow and reap steps. Use this `FORCE_CONSTANTS_3RD` file along with `FORCE_CONSTANTS_2ND` and `CONTROL` to perform a ShengBTE run.

Running ShengBTE:

It is time to run ShengBTE. Gather the `CONTROL, FORCE_CONSTANTS_2ND` and `FORCE_CONSTANTS_3RD` files into a single directory. Examine the `CONTROL` file. It stores all input parameters for the ShengBTE calculation. The ngrid(:) parameter defines ShengBTE integration grid. By default it copies the q-point grid from CASTEP, which is often insufficient to converge the ShengBTE results, hence higher values should be tested. To run ShengBTE simply type ‘`ShengBTE`’ in the terminal.

TASK 3: Make a quick check how the ngrid(:) parameter changes the phonon density of states. Start by running a calculation with the default 2 2 2 grid and plot the `BTE.pdos` file (e.g. `xmgrace -nxy BTE.pdos`). Then kill ShengBTE with Ctrl+C, if it is still running, change ngrid(:) to 3 3 3, run ShengBTE again and plot the PDOS once the `BTE.pdos` file is updated. Notice the difference.

ShengBTE can compute the lattice thermal conductivity at different temperatures. This is specified in the `&parameters` block in the `CONTROL` file. The default parameter `T=300` gives the temperature only at T=300 K. For example, one can investigate the range between 300 and 1000 K in steps of 100 by replacing:

`T=300`
with
`T_min=300`

```
T_max=1000
T_step=100
```

Results for each temperature are written in a separate directory. The converged scalar value of the lattice thermal conductivity is given as the last line in the `BTE.kappa_scalar` file.

TASK 4: Compute the lattice thermal conductivity of NbFeSb between 100 and 900 K is steps of 200. This task should take approximately 1 hour on a single core. Fortunately, ShengBTE can be run in parallel:

```
mpirun -np <num_cores> ShengBTE
```

where <*num_cores*> is an integer showing the number of cores.

Once the calculation is ready plot the lattice thermal conductivity against the temperature.

Hint: You can quickly plot the data using the following command:

```
for i in {100..900..200}; do  echo -n "$i "; tail -1 T"$i"K/BTE.kappa_scalar| awk
'{print $2}'; done| xmgrace -
```

Compare your results for NbFeSb with the literature. If there is a discrepancy how can you improve your estimate?

Additional information:

Hints and tips for thirdorder CASTEP calculations:

- Use `write_checkpoint: none` in the `.param` file. Otherwise, the process of writing hundreds of checkpoint files to the hard drive will slow down the calculation process.

- It is possible to reuse a single checkpoint file for each of the runs. This should save you a couple of hours. For that purpose, generate a checkpoint file from one of the runs and place the file in the root directory where your input files are placed. Then add `reuse : ../../seedname.check` to your `.param` file in the root directory and either run once again thirdorder_castep.py in SOW mode to paste the edited `.param` file to all subdirectories or copy and paste it manually.

- If you don't want to generate the pseudopotentials at the start of each run, you can add the following block to the end of the `.cell` in the root directory:

```
%BLOCK SPECIES_POT
In ../../In_C17_PBE_OTF.usp
As ../../As_C17_PBE_OTF.usp
%ENDBLOCK SPECIES_POT
```

Please note that you need to edit the elements and the name of the pseudopotentials in accordance with your system.

ShengBTE CONTROL file parameters

ShengBTE use values of epsilon=1 (dielectric tensor) and born=0 (Born effective charge tensor) by default. If needed these values can be changed in the `CONTROL` file by adding a few lines to the `&crystal` block.

Epsilon values can be obtained with OptaDOS (for more information, please check this tutorial). Here is an example of how to format the epsilon data in `CONTROL`:

```
        epsilon(:,1)=19.643 0.00 0.00,
        epsilon(:,2)=0.00 19.643 0.00,
        epsilon(:,3)=0.00 0.00 19.643,
```

where this block needs to be placed in `&crystal`.

For polar compounds Born charges can be calculated in CASTEP. For that purpose add "CALCULATE_BORN_CHARGES: true" to the .param file. Born charges should be then given at the end of .castep. Again place the data in the &crystal block in CONTROL, i.e.:

```
        born(:,1,1)=2.67810 0.00 0.00,
        born(:,2,1)=0.00 2.67810 0.00,
        born(:,3,1)=0.00 0.00 2.67810,
        born(:,1,2)=-2.67558 0.00 0.00,
        born(:,2,2)=0.00 -2.67558 0.00,
        born(:,3,2)=0.00 0.00 -2.67558,
```

where the second digit in the brackets represents the type of the atom. For example, if there are 3 species present in the unit cell, 3 extra lines need to be added in addition to the example above. These have the following format born(:,i,3), where i=1, 2, 3 for x, y and z.

For more information on the keywords present in CONTROL and the output generated by ShengBTE, please see: https://bitbucket.org/sousaw/shengbte/overview

Limitations of the CASTEP interface:

- Spin-polarised calculations are not supported at the moment. Spin values will not be included in the supercell files.
- The initial <seedname>.cell file MUST be in the following format:
    - Lattice parameter, Cell contents AND THEN everything else.
- Only fractional coordinates are supported. Use only fractional coordinates.