CASTEP convergence automation tool 1.0 User Guide

Simone Sturniolo

November 2, 2018

1 Introduction

The CASTEP convergence automation tool (CASTEPconv) is a Python script designed to automate the process of calculating the convergence of energy and forces in DFT calculations with CASTEP, as a function of the cutoff energy and the number of kpoints used. It works with Python 2.6 or higher. Given a .cell file, the script is able to create a set of folders containing input files for the required simulations, run the jobs, and process the output to produce tabulated ASCII data files and graphs. A .param file can be introduced as well to define any additional options for the DFT calculation, while a .conv file can be used to provide further options for the automated convergence calculation itself.

2 Installation

To install the script on Linux or CygWin just open a terminal, navigate to the folder containing the files, and type:

```
user@machine:~$ pip install . --user
```

If `pip` is not available on your system, this alternative method can be used, but it's advised to avoid it unless necessary:

```
user@machine:~$ python setup.py install --user
```

CASTEPconv depends on a number of external Python libraries. Installing with `pip` will handle these dependencies automatically. The alternative is to install them by hand. The dependencies are `numpy`, the main numerical library for Python (often coming pre-installed in many distributions of the language) and the Atomic Simulation Environment, or ASE. This can be found in the Python Package Index or downloaded and installed from its repository at `https://wiki.fysik.dtu.dk/ase/index.html`.

3 Usage

CASTEPconv requires only a .cell file to be present in the folder when it's used. Considering a file named "<*seedname*>.cell", where <*seedname*>represents the name of the job, the syntax to run the convergence job is simply

```
user@machine:~$ castepconv.py [options] <seedname>
```

while in the folder where the file is kept. If one wants the convergence DFT simulations to have additional parameters (e.g. redefine the convergence criteria for SCF iterations, add dispersion correction etc.), a <*seedname*>.param file can be added in the same folder with these options. To control the convergence job itself, instead, a new file, <*seedname*>.conv, is needed.
Options before the seedname can be used to quickly override settings in the .conv file. At the moment only one option is accepted, to override the **convergence_task** keyword (see "Options" section).

4 Fine grid convergence (added in version 0.9.5)

Starting from version 0.9.5, besides cut-off energy and k-point grid, the possibility to converge the fine Gmax parameter has been added to CASTEPconv.
Fine Gmax is a different way to express what is also known as 'fine grid' in CASTEP, namely a finer spatial grid than the one on which the electronic wavefunctions are calculated, used mainly to project the electronic density in the pseudopotential region. This grid size can be defined in CASTEP either as a multiple of the size of the standard grid (which, in turn, is controlled by the cut-off) by using the parameter *fine_grid_scale* or with a completely independent parameter *fine_gmax* which is fundamentally equivalent to a cut-off radius, except for the fact that it is expressed in terms of a reciprocal wavelength rather than an energy. This parameter can be considered as the radius of a sphere in the reciprocal space, exactly like the cut-off; and since it describes a grid that is finer than the regular one in direct space, this sphere must be *bigger* than the cut-off one in reciprocal space.
To switch between the representation of this radius in terms of a cut-off energy and that as a reciprocal lattice vector, the regular DeBroglie wavelength formula applies:

$$k_{cut} = \sqrt{\frac{2m_e E_{cut}}{\hbar^2}} \tag{1}$$

where E_{cut} is the cut-off energy and all terms are in SI units. CASTEP's default setting is that both the standard grid and the fine grid have in the reciprocal space a radius of $G_{max} = 1.75k_{cut}$, where 1.75 is the default value of both the *grid_scale* and the *fine_grid_scale* parameters. This is therefore the starting value and the lower boundary for fine Gmax at a given cutoff, since the fine grid can not have a smaller sphere than the standard one.
CASTEPconv allows you to perform a convergence calculation on the fine Gmax

parameter, starting from the minimum value at a given cutoff and increasing in steps of your choice. The parameters to do this are described extensively in the next section. It must be noted that, at a difference from what happens with k-point convergence, the chosen value of cut-off is NOT irrelevant to the result of this convergence: therefore you might want to perform a regular convergence first and only when you have a good guess of what the final cut-off to be used is proceed to perform a second convergence job on fine Gmax. It is possible to have some choice in the cut-off value used for the fine Gmax convergence as well. If you still have doubts about how to proceed, follow through the example in the Getting Started section to try it first hand.

5 Convergence parameters

The syntax of the .conv file is similar to the one of the CASTEP .param file:

```
<parameter name 1>:    <parameter value 1>
<parameter name 2>:    <parameter value 2>
...
```

The accepted parameter names are listed in the subsections below.

5.1 String parameters

convergence_task: Describes the task that is required from the convergence script. This can be INPUT (creation of input files and folders), INPUTRUN (same as INPUT, plus actually runs the jobs), OUTPUT (processes some already finished calculations and creates output files), ALL (does all of the previous things, waiting for the jobs to finish before processing the output) and CLEAR (deletes all files from previous jobs).
Default is INPUT.
Legal values: INPUT, INPUTRUN, OUTPUT, ALL, CLEAR

running_mode: Describes the mode in which the calculations should be ran. If PARALLEL, all calculations will be launched at the same time (ideal for job submission on a cluster). If SERIAL, the program will wait for one calculation to finish before the next one begins (and reuse the .check file from the previous calculation as a starting point). In the latter case, all files will be created in a single folder. Default is SERIAL.
Legal values: PARALLEL, SERIAL

output_type: Plotting program for which an output script should be created. Both GNUPLOT and GRACE are supported. Default is GNUPLOT.
Legal values: GNUPLOT, GRACE

running_command: Command line that should be used to run jobs. This should be expressed by replacing the name of the job with the generic token

<seedname>. Default is *castep <seedname>-dryrun*.
Legal values: any string. If it contains the token *<seedname>*, it will be appropriately replaced.

dryrun_command: Command line that should be used to run a simple dryrun. This should be expressed by replacing the name of the job with the generic token *<seedname>*. If present, during input creation a dryrun will be executed, to test the input files and create pseudopotential files that will be then recycled in the successive runs. Default is None.
Legal values: any string. If it contains the token *<seedname>*, it will be appropriately replaced.

submission_script: Only in version 0.9.2 and higher. The filename of a script that needs to be copied inside all folders before execution. This has been included to accomodate the issues with some large supercomputers that might require job submission to take place by means of such a script rather than a single command. In this case, the script will be renamed as the job, without any extensions, and any instances of *<seedname>*inside it will be replaced by the job name as well. So for example by adding such a script (with the proper job configuration details) it would be possible to submit one's job on a system that supports *qsub* by using `qsub < <seedname>` as a *running_command*.
Legal values: any file name without spaces in it.

fine_gmax_mode: Only in version 0.9.5 and higher. This parameter controls the value of the cut-off used when performing fine Gmax convergence. If set to NONE, no fine Gmax convergence will be performed. If set to MIN_CUTOFF, fine Gmax will be performed with cutoff_min as the cut-off. If set to MAX_CUTOFF, cutoff_max will be used instead. Default is NONE.
Legal values: NONE, MIN_CUTOFF, MAX_CUTOFF

5.2 Float parameters

cutoff_min: Minimum value for the cutoff range explored in eV. Default is 400.0 eV.
Legal values: Any positive float.

cutoff_max: Maximum value for the cutoff range explored in eV. Default is 800.0 eV.
Legal values: Any positive float greater than cutoff_min.

cutoff_step: Step between the values of the cutoff range explored in eV. Default is 100.0 eV.
Legal values: Any positive float.

displace_atoms: Displacement in Angstroms to introduce in atom positions - necessary when the cell is equilibrated and it is not possible to converge forces because they are zero. Default is 0.0 Ang.

Legal values: Any float

final_energy_delta: Tolerance on final energy for the estimate of convergence. Default is 0.00001 eV/atom.
Legal values: Any positive float

forces_delta: Tolerance on maximum force for the estimate of convergence. See above. Default is 0.05 eV/Ang.
Legal values: Any positive float

stresses_delta: Tolerance on maximum stress for the estimate of convergence. See above. Default is 0.1 GPa.
Legal values: Any positive float

fine_gmax_min: Only in version 0.9.5 and higher. Minimum value for the fine Gmax range explored in eV. Default is the minimum value for the used cutoff ($3.0625E_{cut}$). It is ignored if fine_gmax_mode is NONE.
Legal values: Any positive float greater or equal to $3.0625E_{cut}$.

fine_gmax_max: Only in version 0.9.5 and higher. Maximum value for the fine Gmax range explored in eV. Default is fine_gmax_min + 3fine_gmax_step. It is ignored if fine_gmax_mode is NONE.
Legal values: Any positive float greater than $3.0625E_{cut}$.

fine_gmax_step: Only in version 0.9.5 and higher. Step between the values of fine Gmax explored in eV. Default is 100.0 eV. It is ignored if fine_gmax_mode is NONE.
Legal values: Any positive float.

5.3 Integer parameters

kpoint_n_min: Minimum value for the k-point range explored. This applies to the shortest side of the kpoint_mp_grid: depending on the size of the other cell parameters, there might be proportionally more k-points along other sides. Default is 1.
Legal values: Any positive integer.

kpoint_n_max: Maximum value for the k-point range explored. Default is 4.
Legal values: Any positive integer greater than kpoint_n_min.

kpoint_n_step: Step between the values of the k-point range explored. Default is 1.
Legal values: Any positive integer.

max_parallel_jobs: Maximum number of parallel jobs to run when in "parallel" mode. Ignored in "serial" mode. Zero means that there is no limit. Default is 0.
Legal values: Any non negative integer (negative values will be ignored).

5.4 Boolean parameters

converge_stress: Apply calculation of stresses to the simulations and then estimate convergence on stresses as well as energy and forces. Default is False.
Legal values: Anything. The word "true", regardless of the case, means the stresses are calculated. Anything else will be interpreted as False.

job_wait: Wait for jobs to be completed before running the next ones. Only makes sense to set to False when the task is INPUTRUN, and one wants to launch all calculations at once. Careful if using serial mode, because in combination with serial_reuse this will likely fail (as the .check files will not be ready in time). Ignored if the task is ALL. Default is True.
Legal values: Anything. The word "true", regardless of the case, means the stresses are calculated. Anything else will be interpreted as False.

reuse_calcs: If results from a previous convergence are present, recycle them when possible. This requires the .conv_tab file from the previous calculation to be unaltered. Default is False.
Legal values: Anything. The word "true", regardless of the case, means the stresses are calculated. Anything else will be interpreted as False.

serial_reuse: When running a serial calculation, reuse the .check file from previous calculations to start your new simulations instead of starting from scratch to speed up SCF convergence. Default is True.
Legal values: Anything. The word "true", regardless of the case, means the serial reuse is employed. Anything else will be interpreted as False.

castep_8plus: Whether the files should be prepared for CASTEP 8.0 or higher. Allows to use certain keywords that will make the output take less space needlessly but also were not available in previous versions. Default is True.
Legal values: Anything. The word "true", regardless of the case, means that input is prepared for CASTEP 8.0 or higher. Anything else will be interpreted as False.

6 Options

At the moment, CASTEPconv allows only for one command-line option. The option *-t* allows one to quickly override the **convergence_task** keyword in the conv file with another task of choice, using one of the following values as argument: *i, ir, o, c, a* standing respectively for Input, InputRun, Output, Clean and All. For example, to quickly clear the previously written files before restarting a calculation one can quickly issue the command:

```
user@machine:~$ castepconv.py -t c <seedname>
```

7 Output

When the calculations are over, CASTEPconv will have produced the following files, stored in the **<*seedname*>_cconv_out** folder.

<*seedname*>_cut_conv.dat: generated in the OUTPUT phase of the run, will contain a tabulated ASCII of cutoff values (in eV), final energies (in eV) and maximum forces (in eV/Ang) for the various calculations ran. The maximum force is the highest modulus $|\mathbf{f}_i|$ between the forces acting on all atoms i in the system. If **converge_stress** has been set to True, a fourth column will contain the value of the maximum stress component acting on the system.

where σ_{ij} is the generic element of the stress tensor.

<*seedname*>_kpn_conv.dat: generated in the OUTPUT phase of the run, it is similar to the above, except that it expresses its quantities as a function of the total number of kpoints in the grid.

<*seedname*>_fgm_conv.dat: only in version 0.9.5 and higher. Generated in the OUTPUT phase of the run, it is similar to the above, except that it expresses its quantities as a function of the fine Gmax, when this type of convergence has been used. Fine Gmax values will be expressed both in eV (first column) and 1/Angstroms (last column); the latter values are useful because they represent the units currently accepted by CASTEP in .param files.

<*seedname*>_cut_conv.<variable extension>: this file is the script meant for generation of graphic output. By default it will be a Gnuplot script (extension .gp). Choosing the appropriate value for the output_type option will replace it with a different format.

<*seedname*>_kp_conv.<variable extension>: plotting file for k-point convergence (see above).

<*seedname*>_fgm_conv.<variable extension>: plotting file for fine Gmax convergence (see above).

<*seedname*>_report.txt: only in version 0.9.5 and higher. This file contains a printout of the same suggestions for the optimal convergence values that will be printed on the standard output, for future reference.

Besides this, CASTEPconv produces a bit of textual output to suggest which minimum values of cutoff, k-point grid and in case fine Gmax might be the best. This is done by taking the difference between final energy, maximum force, and if

required maximum stress, for successive values and comparing it with a tolerance which, by default, is equal to the tolerance assumed in CASTEP calculations on the same quantities. It should be kept to mind that this is only a rule of thumb method, and that it needs to be considered only as an indication - by no means this is meant to be always the correct answer. Different simulations will also require different convergence criteria. Whenever no convergence is found, or the values examined are too small to be sure of their reliability, a warning message is issued.

8 Getting Started

In this section we'll go through a quick tutorial to getting started with CASTEP-conv, by using the silicon example provided with the code. Even when the example includes running calculations, they should be fairly simple even for a desktop computer. If you are using an especially old or slow machine, please skip those parts.
In order to run the example, just enter the console, move to the *castepconv/example* folder and edit the *Si.conv* file with a text editor. Replace the line

```
running_command: castep.serial <seedname>
```

with whatever the name/path of castep on your system is (remember to keep the <*seedname*>part untouched though). After doing that, you can just use the command

```
user@machine:~$ castepconv.py Si
```

to run the convergence. Just sit back and wait. At the end of the process, after a few minutes, some output files should have been produced. The ones of interest are *Si_report.txt, Si_cut_conv.dat, Si_cut_conv.gp, Si_kpn_conv.dat* and *Si_kpn_conv.gp*. The two .dat files contain tabulated results for energy and forces by varying, respectively, cutoff and k-point grid; the two .gp files are scripts for Gnuplot plotting of the same data sets. If you have gnuplot installed they can be visualized for example with:

```
user@machine:~$ gnuplot Si_cut_conv.gp
```

If you plot the data, you will notice that the line for the "force" is completely flat. This happens because the *Si.cell* file used describes a perfectly equilibrated structure, and in a system in equilibrium, forces are always zero. In order to change this we need to upset this equilibrium a bit. To do so go back to editing the *Si.conv* file and uncomment (remove the hash, #, from the beginning) the line

```
displace_atoms: 0.05
```

which will shift all atoms by 0.05Å in a random direction, enough to give rise to non zero forces. Additionally, if you read the report on the suggested convergence values, you'll see that the convergence with final energy hasn't been

achieved properly. You can try to overcome this both by changing the maximum values for cutoff and k-points and by increasing the tolerance on final energy in the *Si.conv* file. The first can be done by uncommenting the lines

```
cutoff_min:       200
cutoff_max:       1000
cutoff_step:        200
kpoint_n_min:     1
kpoint_n_max:     9
kpoint_n_step:    2
```

which give you some suggested values to start with (though of course you can fiddle with them if you think so). The second instead is done by uncommenting

```
final_energy_delta:       0.001
```

which will raise significantly the tolerance from its original value of 10^{-6} eV/atom. Now rerun the calculation, and when prompted, answer to overwrite the old files. At the end you will be able to see the convergence of forces as well. By following the suggestions given as comments in the *Si.conv* file you'll be able to attempt other test as well - for example enable stress convergence.
If instead you'd like to run everything by hand and let CASTEPconv handle only the input generation and the output postprocessing, that is possible too. First let's clean up the current folder structure by issuing the command

```
user@machine:~$ castepconv.py -t c Si
```

which replaces the current task (*all*) with *clean*, namely, delete all generated files and folders. Then edit the *Si.conv* file and uncomment the following lines:

```
running_mode: parallel
```

and either replace *convergence_task* with *input* and run normally, or run with the option *-t i*. In this way, a structure of folders will be generated, containing the starting files for the various points of the convergence calculations. This could be done with the *serial* running mode as well, but in that case everything would be placed in a single folder and you would need to run the tasks in a specific order for the calculations to succeed (as every calculation in that case reuses the results from the previous one as a starting point in order to improve its performance). After generating the folders, visit each one of them and run the CASTEP calculation normally. When all calculations are finished, go back in the parent folder and either change *convergence_task* with *output* and run normally or run with the option *-t o* in order to get the output postprocessed - as an outcome, you will get the same data files and plots as in the first run.
Finally, let's consider how to run a fine Gmax convergence job. When running this kind of convergence, it is important to use a fully converged cutoff value. If you followed the instructions given above, you should have found that 400 eV make a good enough cutoff value. Now *comment out or delete* all lines defining the range for cutoff and kpoints and scroll down the *Si.conv* file until the end. The lines defining the fine Gmax convergence are here. Uncomment them. They are split in two groups:

- A series of lines redefining a new range for both cutoff and k-points of one point each - this is done for convenience, in order to avoid having to repeat needless calculations;
- The instructions defining the actual testing range for fine Gmax. Please note that since *fine_gmax_mode* is set to *min_cutoff*, the value of the minimum cutoff as defined in the above lines (i.e. 400 eV) will be used. Of course since in this example the same value is used for both minimum AND maximum cutoff, using *max_cutoff* wouldn't have changed much. The advice, however, is to use *max_cutoff* whenever you are running a convergence where you are not yet sure of what the converged cutoff value is either.

A good thing could be to copy the *Si.cell, Si.param* and *Si.conv* files into a new folder and run the convergence anew, so that you don't delete your previous results. Wait for the convergence to finish running and that's it! You're done. You can check a plot and the usual data about the fine Gmax convergence by reading the *Si_fgm_conv.gp* and *Si_fgm_conv.dat* files. If you used stress convergence, there will also be two *Si_fgm_str_conv.** files.