Starting with ONETEP

Arihant Bhandari[1], Rebecca J. Clements[1] and Jacek Dziedzic[1]

[1]University of Southampton

March 27, 2021

Obtaining a copy of ONETEP

If you are a collaborator of one of the members of the ONETEP Developers Group (ODG), you should have received a tarball of a personalised ONETEP copy once you have signed an academic licence agreement. Create a new directory, unpack the tarball there, and you're done.

If you plan to be a ONETEP contributor, you will be using the official ONETEP BitBucket repository, located at `https://bitbucket.org/onetep/onetep`. Create a BitBucket account with your university email address. Contact your supervisor and ask to be added to the `Contributors` group in the ONETEP project to get access to the repository. Read the relevant sections of the `contributing` document, found at `https://bitbucket.org/onetep/onetep/src/master/CONTRIBUTING.markdown`. Follow the instructions there to create your own private fork of the main ONETEP repository. There you will also find details of how to contribute any developments you make to the ONETEP code in the future.

Once you have a private fork of ONETEP on BitBucket, you can make a copy of your ONETEP repository on your local machine, using `git clone`. Make sure you have `git` installed on your computer. From your chosen directory, use one of the following commands, which can also be copied from the BitBucket website, by clicking *clone* at the top of your repository webpage:

```
git clone https://<username>@bitbucket.org/<username>/<repo-name>.git
```

or

```
git clone git@bitbucket.org:<username>/<repo-name>.git
```

and type in your BitBucket password. Substitute `<username>` and `<repo-name>` in the above commands with your username and the name of your private fork, respectively.

Changes to your copy of ONETEP

Any changes to the code should be made in your local clone. Once you are satisfied with them, you can commit them, and push them to your private fork. If you want them to become a part of official ONETEP, you should then create a pull request from your private fork to the official respository. Details are described in the *contributing* document, under *Creating a pull request*.

Whether you are a contributor or a user, you might want to update your repository with any latest changes that might have occurred in the official repository. Users might be interested in recent bug fixes or new functionality, contributors will want to update their copy before committing any changes of their own. The procedure for keeping your repository up to date with the official repository is described under *Development within a fork* in the *contributing* document.

Compiling, testing and running ONETEP

Instructions for setting the environment prior to compiling ONETEP, instructions on how to compile ONETEP, how to run quality-check ("QC") tests that will give you confidence in the robustness of your installation are provided separately – look in the `hpc_resources` directory of your ONETEP installation. There you will also find instructions on how to submit jobs on specific HPC facilities.

Creating input files

Go to ONETEP's website, `onetep.org`. Here you will find the *Tutorials* section, which introduces running various kinds of ONETEP calculations. Take a look at some of the input files at the bottom of the page. Input files in ONETEP have the `.dat` file extension. Should any files get downloaded having a `.txt` extension, you will need to rename them to end with `.dat`.

Input files contain keywords, instructing ONETEP on what calculations to run, and to set the parameters needed to run them. Check out the keywords on the webpage `onetep.org/Main/Keywords` to see what they mean. If not specified, most of them have default settings, as listed on the webpage.

The keywords come in different types: `logical`, `integer`, `real`, `text`, `physical` and `block`. Keywords of the type `logical` can have a value of `T` (true) or `F` (false). Keywords that are `integer` and `real` are numbers. Keywords of type `text` are a string of characters (for example a filename). Keywords of the type `physical` refer to physical variables, which come with units such as angstroem, bohr, joule, hartree, etc. A `block` indicates more than one line of input, these are often used for specifying coordinates.

Some of the important keywords to get started are:

- `task` – to choose what main calculation you would like ONETEP to perform, e.g. a single point energy calculation or geometry optimisation. You can run a properties calculation this way, using output files generated from a single point energy calculation or using `task singlepoint` and a separate keyword `do_properties` set to `T`.
- `xc_functional` – to choose how to approximate the exchange-correlation term in the Kohn Sham DFT energy expression.
- `%block lattice_cart` – to define the dimensions of the simulation cell.
- `%block positions_abs` – to define the atomic positions in Cartesian coordinates.

As can be seen from the example input files, all `block` keywords must end with a corresponding `endblock`. Be default all coordinates are in atomic units (bohr). To switch to angstroems, add `ang` in the first line of the block:

```
%block positions_abs
ang
C 16.521413 15.320039 23.535776
O 16.498729 15.308934 24.717249
...
%endblock positions_abs
```

The `species` and `species_pot` blocks detail the parameters of the atoms. Non-orthogonal Generalised Wannier Functions (NGWFs) are used to model the atomic orbitals. In the `species` block, the name we give to each type atom in the system is given first, followed by the element of the atom, its atomic number, the number of NGWFs to use (use -1 for an educated guess) and the radius of each NGWF typically around 8.0-10.0 (in bohr) for an accurate calculation. For instance for carbon you might use:

```
C C 6 4 8.0
```

The `species_pot` block specifies the location of the pseudopotential used for each element of the system. The standard ONETEP norm-conserving pseudopotentials (`.recpot` files) exclude core electrons. Core electrons are included in `.paw` files. Some of these can be found in your repository's `pseudo` directory. A complete database of all pseudopotentials for all elements in the `.paw` format can be downloaded from `https://www.physics.rutgers.edu/gbrv/all_pbe_paw_v1.5.tar.gz`

To continue a calculation if it has run out of computation time, use the keywords below. The original input must have the `write` keywords, but no

`read` keywords because the files aren't available to read at this stage. Any continuing input files must include the `read` keywords. If the input file name isn't changed upon continuation, the output file will be overwrite with the results of the continuation, so make sure to back up files before continuing.

```
write_denskern T
write_tightbox_ngwfs T
read_denskern T
read_tightbox_ngwfs T
```

If you are running an ensemble DFT (EDFT) calculation you will also need to add
`write_hamiltonian T`
`read_hamiltonian T`
to the above list.