TcESTIME User Manual

Version 2.0 – October 2025

TcESTIME
IS TIME

Group led by Mme. Julia Contreras-García
Laboratoire de Chimie Théorique, Sorbonne Université

This manual provides hands-on examples and usage instructions for TcESTIME and TcESTIMEWeb, including workflows for both Quantum ESPRESSO and VASP.

Repository: `https://github.com/juliacontrerasgarcia/Tcestime`

First Comments

TcESTIME is developed in Python 3 and requires the following packages to operate:

- `numpy`
- `scipy`
- `pandas`
- `matplotlib`
- `scikit-learn==1.4.1` (other versions might fail for the GBR method)
- `onnxruntime`

In addition, TcESTIME depends on the topological analysis of the electron localization function (ELF), which must be performed with a stable version of `critic2`. We strongly recommend using the official release distributed by A. Otero-de-la-Roza, available at: `https://aoterodelaroza.github.io/critic2/`, compiled in its default configuration.

The examples and workflows provided in this manual have been tested under:

- Python 3.12 and Python 3.13
- `critic2` version 1.1
- Quantum ESPRESSO versions 6.3–7.5
- VASP version 6.2.1

All examples are fully reproducible using the input files available in the GitHub repository of TcESTIME.

Running TcESTIME

Before running TcESTIME, make sure that Python can locate the main script by setting the environment variable `PYTHONPATH` to the folder containing `TcESTIME.py`. This can be done by executing the following command in the terminal:

```
export PYTHONPATH=/path/to/TcESTIME
```

Once configured, the code can be executed from any directory using the standard syntax described in the following sections.

1 Hands-On: TcESTIME for Quantum ESPRESSO < 6.8

For QUANTUM ESPRESSO (QE, versions < 6.8), TcESTIME estimates the superconducting critical temperature from the topological analysis of the electron localization function (ELF) and the projected density of states (PDOS). The required inputs are the `critic2` output file and the PDOS files generated by QE's post-processing utilities.

The complete workflow is summarized below:

1. **Self-consistent calculation (SCF):** Run the non–spin-polarized ground-state calculation with `pw.x`.
2. **Non-self-consistent calculation (NSCF):** Perform an NSCF run to sample the electronic structure accurately around the Fermi level. The resulting file `nscf.out` should be kept in the working directory.
3. **ELF computation (PP step):** Use `pp.x` to generate the ELF on the real-space grid:

```
&INPUTPP
 prefix = 'system'
 outdir = './'
 plot_num = 8
 filplot = 'system.ELF.dat'
/
&PLOT
 iflag = 3
 output_format = 6
 fileout = 'system.ELF.cube'
/
```

   This produces `system.elf`, which can be analyzed by `critic2`.

4. **PDOS calculation:** Use `projwfc.x` to obtain the projected DOS files (`*.pdos.*`) for each atomic orbital.

```
  &projwfc
    outdir = './',
    prefix = 'system',
    ngauss =1, degauss = 0.005,
    DeltaE = 0.001
    kresolveddos=.false.
    filpdos='system'
    filproj='system.dat'
/
```

5. **ELF topological analysis (Critic2):** Run `critic2` on the ELF grid to locate critical points. The following parameters are recommended:

```
  crystal system.ELF.cube
  load system.ELF.cube
  AUTO CPEPS 0.3 NUCEPSH 0.6
```

   This step produces the file `critic.out`, required by TcESTIME. It is possible to decide whether critic2 should do analytical (default) or numerical derivatives to find the critical points. If the user decided to change the default mechanism, the keyword `NUMERICAL` should be added at the end of the second line.

Execution

Once all required files are available, run TcESTIME as follows:

```
python3 TcESTIME.py critic.out --fit SR4
```

The flag `--fit` specifies the analytical or machine-learning model used for the estimation of T_c. If not provided, TcESTIME automatically selects NN by default.

Thus, if `--fit` is omitted, the following command is equivalent:

```
python3 TcESTIME.py critic.out
```

By default, TcESTIME searches for all PDOS files (`*.pdos.*`) in the current working directory. If the PDOS files are located elsewhere, specify their directory using:

```
--dpdos /path/to/pdos_dir
```

Example Output

If the above procedure is followed, and the example provided in the official repository (`Examples/QE/QE_1/`) is used, the expected output should appear as follows:

```
python3 /path/to/TcESTIME.py critic.out --fit SR4
Fermi energy automatically read from 'nscf.out': 24.1872 Ry
Fermi energy : 24.187 Ry
H_DOS: 0.784
Molecularity index:  0.79
Found one-dimensional periodic network at isovalue  0.63
Found two-dimensional periodic network at isovalue  0.63
Networking value: 0.6
H_f: 0.900
Fit for the estimation of Tc: SR4
Tc^{SR4} = 574.7 * phi * (HDOS * Hf^3)^{1/2}
Critical temperature: 260.75K

Time: 0.50 s
```

This output corresponds to the example system provided in the GitHub repository (`Examples/QE/QE_1/`). It confirms that all descriptors (ϕ, ϕ^*, `H_f` and `H_DOS`) were correctly extracted from the ELF and DOS data, and that the symbol regression model SR4 was successfully applied. For this particular material, $CaYH_{20}$, the reference critical temperature is 250 K.

2 Hands-On: TcESTIME for Quantum ESPRESSO ($6.8 \leq$ QE ≤ 7.5)

For Quantum ESPRESSO versions between 6.8 and 7.5, TcESTIME can be applied following the same workflow as described in the previous section. However, it is important to note that these QE releases contain a known issue in the computation of the electron localization function (ELF) when performing non–spin-polarized calculations. This bug, reported by our group in the official Quantum ESPRESSO repository (`https://gitlab.com/QEF/q-e/-/issues/819`), affects the normalization of the ELF scalar field, leading to unphysical ELF values in the generated `.cube` file.

To correct this issue, it is necessary to rescale the ELF cube before performing the topological analysis with `critic2`. For this purpose, a utility script named `cube_tool.py` is provided in the `tools/` folder of the TcESTIME GitHub repository. This tool allows automatic rescaling of the ELF field and generation of a corrected cube file:

```
python3 cube_tool.py elf-rescale system.ELF.cube system_ok.ELF.cube
```

The corrected ELF cube (`system_ok.ELF.cube`) should then be used as input for `critic2` to locate the ELF critical points, as in the standard workflow:

```
crystal system_ok.ELF.cube
load system_ok.ELF.cube
AUTO CPEPS 0.3 NUCEPSH 0.6
```

After this correction, the workflow proceeds identically to that described in the previous section (SCF, NSCF, PDOS, and TcESTIME execution). An illustrative example of this corrected workflow is provided in the GitHub repository under `Examples/QE/QE_2/`, which includes all necessary input and output files to reproduce the procedure described above.

3 Hands-On: TcESTIME for VASP

In the VASP workflow, TcESTIME requires as its main input the output file generated by `critic2` analyzing the electron localization function (ELF) scalar field and the projected density of states (PDOS) obtained from a separate DOS calculation. The overall procedure is as follows:

1. Perform a standard non–spin-polarized self-consistent field (SCF) calculation, ensuring that the `ELFCAR` file is generated (`LELF = .TRUE.`).
2. Once the SCF calculation is complete, perform a DOS calculation around the Fermi level. It is recommended to use `LORBIT = 10` and define an energy window of approximately ± 0.5 eV around E_F, with at least 1000 points to ensure smooth sampling.
3. Run the stable version of `critic2` on the ELF scalar field to identify its critical points. Note that one must not change the name of the ELF file (e.g. to `ELFCAR_sys`), as critic2 will yield incorrect results. For this step, we recommend using the following parameters:

```
crystal ELFCAR
load ELFCAR
AUTO CPEPS 0.3 NUCEPSH 0.6
```

   As with Quantum ESPRESSO, the user can decide to change the algorithm to compute the derivatives using an extra keyword (see Sec. 1).

4. Execute TcESTIME using the `--code vasp` flag and specifying the critic2 output file:

```
python3 TcESTIME.py system.critic.out --code vasp --fit NN
```

When invoked with `--code vasp`, TcESTIME automatically assumes that:

- the SCF output files (ELFCAR, etc) are present in the current working directory, and
- a subdirectory named `dos/` exists, containing the DOS calculation output (typically `DOSCAR` and `PROCAR`).

If these conditions are met, the program will extract all relevant descriptors directly from the available data, without requiring manual specification of the DOS path or hydrogen DOS value.

Execution

Once all required files are available, run TcESTIME for VASP as follows:

```
python3 TcESTIME.py critic.out --code VASP
```

For VASP calculations, the flag `--fit` is optional and automatically set to `NN`, which is the only available and supported model for this workflow. Specifying `--fit` explicitly is therefore unnecessary; however, if it is provided, its value must be `NN`. Any other option (e.g., `SR4`, `GBR`) will result in an execution error, as no other models have been trained for VASP data.

By default, TcESTIME searches for the SCF output files in the working directory, and expects a subdirectory named `dos/` containing the DOS calculation outputs (`DOSCAR` and `PROCAR`). If the DOS files are located elsewhere, specify their directory using:

```
--dpdos /path/to_dos_dir/
```

where the SCF output files are expected to be, while the DOS output files are expected to be in

```
--dpdos /path/to_dos_dir/dos
```

Example Output

If the above procedure is followed, and the example provided in the official repository (`Examples/VASP`) is used, the expected output should appear as follows:

```
python3 /path/to/TcESTIME.py critic.out --code VASP
Fermi energy automatically read from DOSCAR file: 14.771 eV
H_DOS: 0.589
Molecularity index:  0.72
Found one-dimensional periodic network at isovalue  0.63
Found two-dimensional periodic network at isovalue  0.63
Networking value: 0.55
H_f: 0.909
Fit for the estimation of Tc: NN
Tc^{NN}_{VASP} predicted with neural network ensemble trained on VASP data
Critical temperature: 220.78K

Time: 0.04 s
```

This output corresponds to the example system provided in the GitHub repository (`Examples/VASP`). It confirms that all descriptors (ϕ, ϕ^*, `H_f` and `H_DOS`) were correctly extracted from the ELF and DOS data, and that the neural network model was successfully applied. For this particular material, $CaYH_{20}$, the reference critical temperature is 250 K.

4 Hands-On: TcESTIMEWeb

TcESTIMEWeb provides a web interface available at:
`https://lct-webtools.sorbonne-universite.fr/tcestime/`

Usage Steps

1. Upload the `critic2` output and (optionally) PDOS files.
2. Choose the computational source (VASP or QE).
3. Select the desired fit model.
4. Click **Run** to obtain T_c and descriptors.

The results can be downloaded and include: ϕ, ϕ^*, H_f, H_{DOS}, and T_c.