2FAST2CAR - User Manual

Oscar Gayraud, Hugo Geindre, Frédéric Guégan

April 18, 2025

1 Introduction

1.1 FASTCAR concept

FASTCAR is designed as a complementary tool to CREST, enabling a fully automated exploration of the conformational degrees of freedom within reaction profiles with results optimised in the level of theory of your choice. FASTCAR will use CREST to generate conformers of any geometry corresponding to a stationary point you feed it and reoptimised the CREST results with the method and basis set specified. It uses an iterative approach to ensure the completeness of the result and can run a range of different calculations on the conformers obtained.

1.2 Foreword and new stuff

2FAST2CAR is the second version of the program FASTCAR (see *Phys. Chem. Chem. Phys.* **2024**, *26*, 25780-25787). In this, we added some functionalities:

- now accepting output files from all packages compatibles with cclib jobs;
- running by default in iterative mode: one a first run is complete (CREST search, geometries pruning, DFT calculations and second stage pruning), a new search is initiated starting from the lowest energy conformer at the DFT level;
- post-SCF analyses: user may require the calculation of conceptual DFT indices (finite difference approximation), as well as the production of wfn/wfx files for further jobs (QTAIM, ELF...);

and some further things are underway.

1.3 Installation

1. Download all the files in the `src` folder from github and put them in a directory of your choice;
2. Execute the `configure.py` script to update the path to the fastcar directory, which is hardcoded in all python scripts;
3. Update the file `config.txt` with the paths where the quantum chemistry packages that you plan to use are located on your cluster;
4. You can also add the program folder to your path to be able to easily call the starting script (`sparkplug.py`);
5. Make sure you have all the dependencies needed (see following subsection).

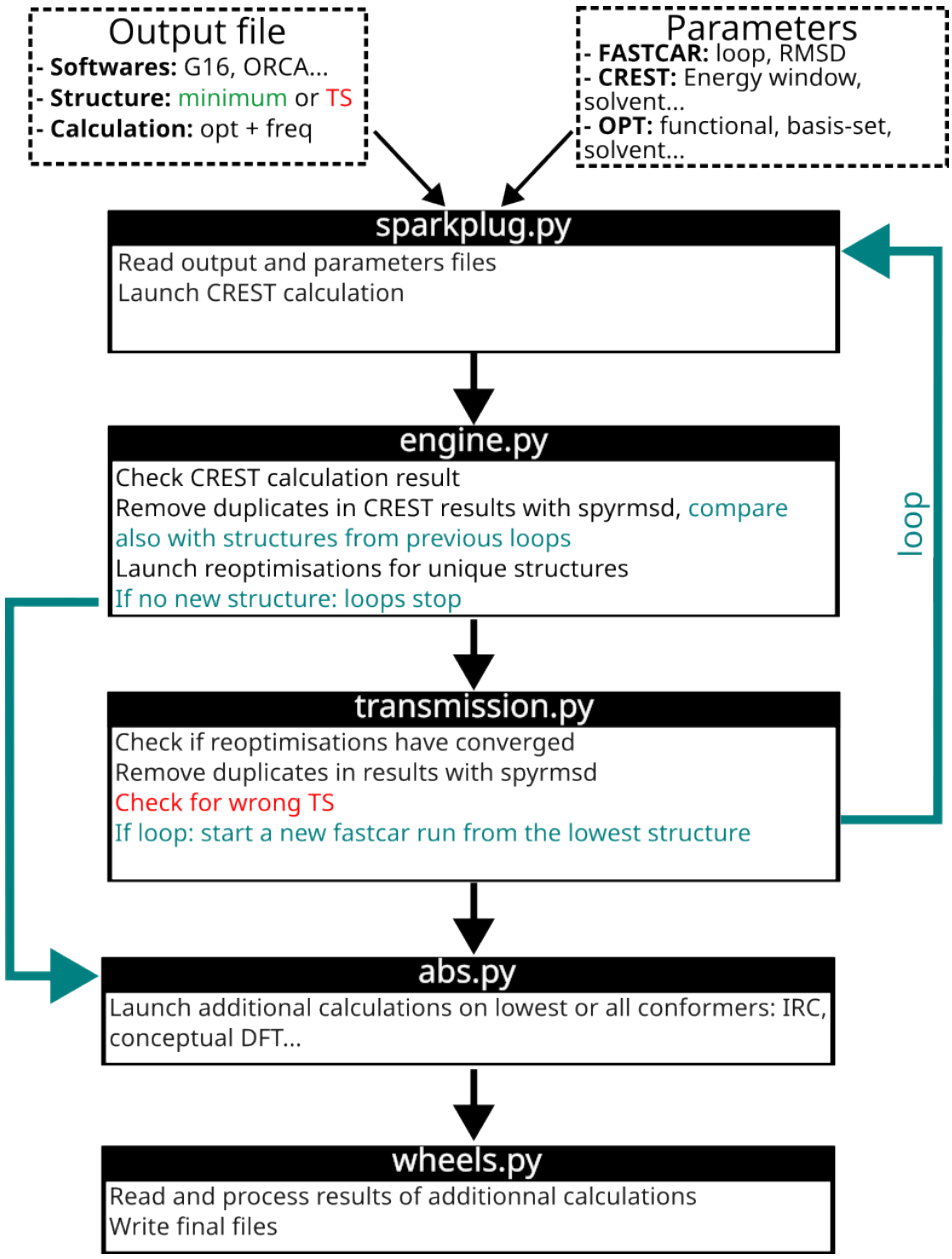
1.4 Dependencies

FASTCAR is currently designed to operate under Python 3 (tested against Python 3.8.18). It relies on the use of the following dependencies:

- CREST;
- SpyRMSD;
- cclib;
- python libraries: `os`, `sys`, `re`, `shutil`, `glob`, `subprocess`, `datetime`, `math`, `numpy`, `scipy`, `matplotlib`;

The present implementation is designed to work with Slurm (tested against Slurm 23.11.4).

2 Workflow



3 Typical usage

3.1 Running a fastcar calculation.

In the desired folder, containing the output of a calculation from a package compatible with cclib (minimum or transition state, optimised and with a frequency calculation), the sequence is initiated by calling `sparkplug.py`. This script requires as argument the name of the output file, and uses a second file called `parameters.txt` containing all the run parameters. Its content is divided into 4 sections, discussed hereafter.

The file `parameters.txt` is not case sensitive, the different sections are purely cosmetic, FASTCAR only read the keywords and what is written after in the line. Most keywords have a default value that will be used if the keyword is absent from the parameters file, the only mandatory keywords are: `method` and `basis set`

Do not launch two calculations running at the same time with an identical name (or with one name being included in the other). FASTCAR is monitoring if calculations have ended by searching for the job names so if two calculations have similar names it will mess up with the monitoring and the run will take more time.

3.1.1 FASTCAR parameters.

- `Max loops`: should be an integer or `none`. Define the maximum number of loops after which the FASTCAR calculation will stop even if it has still found new conformers, if it's `none`, no loop is done. Default value: `10`.

3.1.2 CREST parameters.

- `EWIN`: energy window for the conformers selection, in kcal/mol. Default value: 6.
- `CREST version`: this keyword indicates the crest version to be used. User may either provide a specific version (*e.g.* 2.12) or `default` (default version in the python environment of the user). Default value: `default`
- `CREST solvent`: indicates whether a solvent model (using the analytical linearized Poisson-Boltzmann model) should be used during the CREST search.[1] The user should here provide the appropriate xtb keyword, for instance

  ```
  CREST solvent --alpb THF
  ```

  or indicate `none` if no solvent is to be used. Default value: `none`
- `NCI`: indicates whether the non-covalent biased search should be used. The keyword alone will just activate NCI search, if you put a number as argument it will be used as scaled factor. The argument `none` can also be used to deactivate NCI search. Default value: `none`
- constraints: constraints on bonds, angles and dihedrals can be enforced. Each constraint should be written on a separate line starting by the appropriate label (`bond constrained`, `angle constrained`, `dihedral angle constrained`[2]) followed by the number of atoms involved in the constraint.[3] Default value: no constraints;

[1] https://xtb-docs.readthedocs.io/en/latest/gbsa.html
[2] https://gist.github.com/hypnopump/30d6bfdb8358d3a57d010c9a501fda56
[3] https://crest-lab.github.io/crest-docs/page/examples/example_3.html#{}sampling-of-non-covalent-complexes-and-aggregates-

- `Force constant`: provides the value of the force constant used in the constraint. Default value: no force constant;
- `Crest time limit`: Define the time limit that will be used in slurm for the crest calculations. Format is the slurm format [dd-]hh:mm:ss (for instance 02-05:30:00 is 2 days, 5 hours and 30 minutes). Default value: `48:00:00`.

3.1.3 RMSD parameters.

- `RMSD threshold`: the RMSD threshold for the rejection of duplicate geometries. If two arguments are given the first value is used when comparing CREST structures and the second is used when comparing reoptimised structures. If only one is given it will be used for the CREST structures and the default value will be used for reoptimised structures. Default value (for both): `0.1`.

3.1.4 OPT parameters

- `Software`: the software that you want to use for the calculations. The name must match the names of the input file and submission script models in the Models/ folder (see section 4.4) If the keyword given is `same` FASTCAR will use the same software that was used for the calculation that was provided as a starting point. More informations on the syntax in the section 4. Default value: `same`.
- `Method`: the calculation method that will be used. It will be copied as it is in the input file so it should be a keyword accepted by the software you are planning to use.
- `Basis set`: the basis set to use. It will be copied as it is in the input file so it should be a keyword accepted by the software you are planning to use.
- `Dispersion`: the dispersion model to be used. It will be copied as it is in the input file so it should be a keyword accepted by the software you are planning to use. If `none` is specified, no dispersion model will be used. Default value: `none`.
- `Opt solvent`: the solvent model to use. It will be copied as it is in the input file so it should be a keyword accepted by the software you are planning to use (*e.g.* `solvent scrf=(solvent=methanol)` for G16 or `CPCM(water)` for ORCA). If `none` is specified, no solvent will be used. Default value: `none`.
- `Time limit`: Define the time limit that will be used in slurm for the optimisation calculations. Format is the slurm format [dd-]hh:mm:ss (for instance 02-05:30:00 is 2 days, 5 hours and 30 minutes). Default value: `48:00:00`.
- `Nodes excluded`: The argument of this keyword can be `none` (if no exclusion) or any valid argument for the slurm sbatch option `exclude`: a single node identifier, a list or a range. Default value: `none`.
- `TS screening`: This keyword can be used to chose how FASTCAR will compare the transition states obtained after reoptimisation with the starting transition state to decide if they describe the same reaction. The argument of this keyword can be `scalprod` or `activats` (see section 6 for more informations). Default value: `scalprod`.
- `Guided opt`: . Default value: `scalprod`.

3.1.5 Additional calculations parameters

- `additional calculation`: The following additional calculations can be carried out on the lowest energy structure or on all conformers (see `scope` keyword): `single-point` (a single point calculation), `wfx` (a single point calculation with wfx files generated), `reopt` (geometrical optimisation), `reopt` (geometrical optimisation), `irc` (IRC forward and reverse calculation, only for TS), `cdft` (compute several conceptual dft descriptors), `nbo` (Natural Bond Orbital analysis with NBO7), `none` (no additional calculation will be performed). Several additional calculations can be requested by writing several keywords on the same line, separated by a space. Default value: `none`.

- `scope`: Determine on which structures the additional calculations will be performed. Either `lowest` (only on the lowest energy conformer) or `all` (on every unique conformers found). Several different scopes can be specified (separated by spaces) if several additional calculations are requested. If less values of scopes are specified than additional calculations, the additional calculations not matched with specific values of scope will use the last one. Default value: `lowest`.

- `additional method`: the method that will be used to perform the additional calculations. Syntax similar to the method of the OPT parameters section. Several methods can be specified (separated with spaces), syntax working like `scope`.

- `additional basis set`: the basis that will be used to perform the additional calculations. Syntax similar to the basis set of the OPT parameters section. Several basis set can be specified (separated with spaces), syntax working like `scope`.

Some examples of additional calculations with different scope, method and basis set to clarify:

```
Additional calculation reopt irc
Scope all lowest
Additional method MP2 b3lyp
Additional basis set aug-cc-pVTZ 6-311++G(d,p)
```

will perform a reoptimisation for every conformers at MP2/aug-cc-pVTZ level and an IRC calculation for the lowest conformer at b3lyp/6-311++G(d,p) level.

```
Additional calculation nbo irc
Scope all
Additional method b3lyp
Additional basis set 6-311++G(d,p)
```

will perform a NBO calculation for every conformers at b3lyp/6-311++G(d,p) level and an IRC calculation for every conformers at b3lyp/6-311++G(d,p) level.

```
Additional calculation single-point reopt irc
Scope lowest
Additional method CCSD(T) MP2 b3lyp
Additional basis set aug-cc-pVTZ 6-311++G(d,p)
```

will perform a single-point for lowest conformer at CCSD(T)/aug-cc-pVTZ level, a reoptimisation for lowest conformer at MP2/6-311++G(d,p) level and an IRC calculation for lowest conformer at b3lyp/6-311++G(d,p)

3.2 Parameters file example

Hereafter is reproduced an example of a FASTCAR parameters file.

```
-------------------------------------------------
----------------FASTCAR parameters---------------
-------------------------------------------------
Max loops 10

-------------------------------------------------
----------------CREST parameters-----------------
-------------------------------------------------
CREST version default
CREST solvent --alpb THF
EWIN 4
NCI
Bond constrained 13 12
Angle constrained 11 17 18
Dihedral angle constrained 2 3 7 9
Force constant 1

-------------------------------------------------
-----------------RMSD parameters-----------------
-------------------------------------------------
RMSD threshold 0.3

-------------------------------------------------
-----------------OPT parameters------------------
-------------------------------------------------
Software g16
Method  b3lyp
Basis set  6-311++G(d,p)
Dispersion  empiricaldispersion=gd3
DFT solvent  SCRF=(Solvent=THF)
Time limit 02-05:30:00
Nodes excluded 01-08

-------------------------------------------------
------------Additionnal calculations-------------
-------------------------------------------------
Additional calculation reopt irc
Scope lowest
Additional method MP2 b3lyp
Additional basis set aug-cc-pVTZ 6-311++G(d,p)
```

3.3 Results

3.3.1 File `summary.log`

In this file you can find a text resume of all the steps of the calculation. For every loop you have informations on the starting structure, the CREST parameters, the number of conformers found by CREST and how many duplicates, the optimisation parameters, the results of the optimisations.

3.3.2 File `structures.log`

In this file you have a list of all optimised structures found divided into different categories:

- **TS/Min unique:** all the unique conformers with their energies (and the first frequency for the TSs)
- **Duplicates**: all conformers which are identical to another one, with their energies (first frequencies for TS), the names of the structure they are identical to and the values of the RMSD to this structure.
- **Garbage:** for minima, all conformers which have a negative first frequency or for which the optimisation has failed. For TS, all conformers which have a positive first frequency or which are describing another TS or for which the optimisation has failed.

3.3.3 Files `pruningCREST.log` and `pruningOPT.log`

One `pruningCREST.log` and one `pruningOPT.log` are created for each loop, with _n added to the name, n being the number of the loop. The file gives a summary of the pruning with the threshold used, a list of the unique conformers and a list of the duplicates with the RMSD values.

3.3.4 `dashboard.py` module

The `dashboard.py` module allow to visualize the calculation results or to create other text files containing the results of the calculation. It need to be executed from the directory you created for the calculation, there are 8 different options:

1. **OPT energies vs CREST energies:** energies of the optimised structures in function of the energies of the CREST structures used as starting point for the optimisation.
2. **OPT RMSD vs OPT energies:** RMSD between optimised structures in function of their difference in energy.
3. **OPT RMSD vs CREST RMSD:** RMSD between optimised structures in function of the RMSD between the CREST structures used as starting point for the optimisation.
4. **CDFT:** if a CDFT additional calculation has been performed, it will produce a text file named `cdft.log` with the values of global descriptors (ionisation potential, electronic affinity, chemical potential μ, chemical hardness η, electrophilicity index ω) and local descriptors (Fukui functions f^+, f^- and dual descriptor Δf. s^+, s^- and Δs. $\Delta\rho_{\text{elec}}$, $\Delta\rho_{\text{nuc}}$.)
5. **Pruning:** show a list of all duplicate CREST conformers with the following informations: [duplicate conformer number] [duplicate conformer loop number] ([unique conformer number] [unique conformer loop number] [rmsd]) and ask to chose which one to show. The chosen conformer will be shown aligned with the identical unique conformer.

6. **Boltzmann:** show the percentage of Boltzmann population due to each loop.

7. **Structures:** show a bar chart with informations on the conformers found at each step of the calculation.

8. **Write energies csv:** write a csv file `calculation_name_nrjs.csv` containing the names of the conformers, their absolute energies in Hartree and their relative energy in kcal/mol.

4 Quantum chemistry packages

Thanks to cclib, FASTCAR is able to read output from most of the common computational chemistry packages. However, by default it is only able to perform calculations with Gaussian and ORCA. This section is intended to explain how FASTCAR is submitting calculations with these packages, to help if you have an issue with calculation submission or if you want to make FASTCAR compatible with a new package.

4.1 Alias

Each versions of a quantum chemistry package must be associated with an alias that will be used by FASTCAR to submit calculations. This alias must be consistent in all the places it used, *e.g.* after the `software` keyword in the parameters file (see section 3.1.4), to give the pathes to the packages in the config.txt file (see section 4.2), in the names of the input and submission scripts models (see section 4.4).

4.1.1 Default alias

If the `software` option in parameters file is set to "same" (which is the default value), FASTCAR will determine which package has been used to produce the output given as a starting point and will use the same package. A default alias for the package must be defined in the `config.txt` file with the syntax:

```
default_{package}={alias of default version}
```

The {package} variable must be the name of the quantum chemistry package as given by cclib.

4.2 Pathes

Absolute pathes to the directories of the different quantum chemistry packages must be specified in the `config.txt` file with the following syntax:

```
path_{alias}={absolute path to the package directory}
```

4.3 Keywords

FASTCAR can submit different type of calculations using the quantum chemistry packages, for each package and each type of calculation, users need to provide the keywords that will be copied in the input file. There are currently 12 types of calculation:

1. `opt`: these keywords are used to optimised the conformers after the CREST calculation in the case of a minimum, it needs to be an optimisation and frequency calculation.

2. `reopt`: these keywords are used when a reoptimisation of all unique conformers is requested in additional calculations, in the case of a minimum.

3. `opt-ts`: these keywords are used to optimised the conformers after the CREST calculation in the of a transition state, it needs to be an optimisation and frequency calculation.

4. `opt-ts-guided`: these keywords are used to optimised the conformers, with an emphasis on some internal coordinates, after the CREST calculation in the case of a transition state, it needs to be an optimisation and frequency calculation. These keywords are used when the `guided opt` keyword is specified in the parameters file.

5. `reopt-ts`: these keywords are used when a reoptimisation of all unique conformers is requested in additional calculations, in the case of a transition state.

6. `freq`: these keywords are used when a simple frequency calculation is needed.

7. `single-point`: these keywords are used when a single point calculation is requested as additional calculation.

8. `wfx`: these keywords are used when a single point calculation with wfx file generation is requested as additional calculation.

9. `irc`: these keywords are used when IRC calculations are requested as additional calculations to perform the IRC calculation.

10. `irc-opt`: these keywords are used when IRC calculations are requested as additional calculations to reoptimised the results of the forward and reverse IRC calculations.

11. `cdft`: these keywords are used when CDFT calculations are requested as additional calculations to compute charges in the system with different models that will be used to compute the descriptors.

12. `NBO`: these keywords are used when NBO calculations are requested as additional calculations.

    Example of all these keywords for gaussian 16 and orca 4 are written in the sections 4.5.1 and 4.5.2, there are also provided by default with the FASTCAR files (but users are allowed to change them at their convenience).

4.4 Models

For each package, there must two files in the `Models/` directory, a model for the input file named {`alias`}`.inp` and a model for the submission file named {`alias`}`.sub`. These files will be used as models for FASTCAR to write input and submission files for each calculation, for that they must contain some specific keywords within braces that will be replaced with specific variables. Two lists of these keywords for input files and submission scripts are given hereafter. The keywords are not mandatory, if a keyword from this list is absent from the model file it will simply be ignored.

4.4.1 General input file model

There are currently 12 different keywords for input files that are recognised by FASTCAR. They can be divided in two categories. First, the basic ones that will simply be replaced with a variable:

1. {cpus}: this will be replaced by the number of cpus requested for each calculation specified in the parameters file.

2. {chk}: this will be replaced with chk=calc_name.chk (calc_name being the name of the calculation submitted) to request a chk file to be written.

3. {keywords}: this will be replaced by the keywords that depend on the type of calculation and on the package, which must be specified in the `keywords.txt` file.

4. {method}: this will be replaced with the keyword for the method specified in the parameters file.

5. {basis-set}: this will be replaced with the keyword for the basis-set specified in the parameters file.

6. {solvent}: this will be replaced with the keyword for the solvent specified in the parameters file.

7. {dispersion}: this will be replaced with the keyword for the dispersion specified in the parameters file.

8. {charge}: this will be replaced with the charge of the system.

9. {multiplicity}: this will be replaced with the multiplicity of the system.

10. {geo}: this will be replaced with the coordinates of the all the atoms of the system, each line being: element x y z.

11. {wfx}: this will be replaced with the name of the wfx file (calc_name.wfx, calc_name being the name of the calculation submitted).

There are also more complex keywords which can be seen as optional customizable blocks in the input. These also have a special keyword within braces, but everything between this keyword and the closing brace will be copied in the input file. There can also be keywords within square brackets in this copied part.

12. {int coord [T] [A1] [A2] [A3] [A4]}: this block will be used in case of guided transition-state optimisation to define the internal coordinates to follow. Everything written between the braces (except from the keyword constraints and the space just after) will be copied in the input file. [T] will be replaced with the type of coordinate (B for bond, A for angle or D for dihedral) [A1], [A2], [A3], [A4] will be replaced with the number of the atoms (only [A1] and [A2] if it's a bond, only [A1], [A2] and [A3] if it's an angle). If there are several coordinates the whole block will be copied as many times as the number of coordinates.

4.4.2 Specific input file models

If you are not able to write what you need in the input file with the keywords given just above, you can write input file models specific to a type of calculation. The name of these models must be {`alias`}_{`calculation-type`}`.inp`, calculation-type must match one of the calculation types specified in the second column of the `keywords.txt` file (opt, opt-ts, freq, irc, ...). If there is a specific model for a type calculation for the requested package, FASTCAR will use this model for this type of calculation, if there is not it will use the general input file model. The keywords specified above can also be used in specific input file models in the same way as in the general input file model.

4.4.3 Submission script model

There are 5 differents keywords for submission scripts that are recognised by FASTCAR, 4 basic ones:

1. {jobname}: this will be replaced with the name of calculation submitted
2. {cpus}: this will be replaced by the number of cpus requested for each calculation specified in the parameters file.
3. {time}: this will be replaced by the time limit requested for each calculation specified in the parameters file.
4. {soft_path}: this will be replaced by the path to the package directory which is specified in the `config.txt` file.

and one complex:

5. {nodex [nodenumbs]}: if there are excluded nodes specified in the parameters file, this block will be copied with [nodenumbs] replaced by the numbers of the nodes.

4.5 Examples

To clarify everything explained earlier in this section, here is a complete example of how it works for the Gaussian 16 and the ORCA 4 packages.

4.5.1 Gaussian 16

The alias for the Gaussian 16 package is g16, so the corresponding line in the parameters file should be:

```
software g16
```

In the `config.txt` there must be a line with the absolute path to the directory where the executable for Gaussian 16 is located:

```
path_g16=/home/pub/software/g16C/g16/
```

and a line to specify that the default alias for Gaussian is g16:

```
default_gaussian=g16
```

Keywords for different type of calculations are specified in the `keywords.txt` file with the syntax `alias;calculation type;keywords`:

```
g16;opt;opt freq=noraman
g16;reopt;opt freq=noraman
g16;opt-ts;opt=(calcfc,ts,noeigen) freq=noraman
g16;opt-ts-guided;opt=(calcfc,ts,noeigen,modredun) freq=noraman
g16;reopt-ts;opt=(calcfc,ts,noeigen) freq=noraman
g16;freq;freq=noraman
g16;single-point;
g16;wfx;output=wfx
```

```
g16;irc;irc=(direction,maxpoints=50,recalc=3,calcfc,LQA)
g16;irc-opt;opt freq=noraman geom=check guess=read
g16;cdft;pop=(NPA,hirshfeld)
g16;nbo;pop=NBO7
```

There also must be two files `g16.inp` and `g16.sub` in the `Models/` directory. The default file `g16.inp` is:

```
%nprocshared={cpus}
{chk}
%mem=5GB
# {keywords} {method} {basis-set} {solvent} {dispersion}

VROUUUUUUUUUUUUUM

{charge} {mult}
{geo}

{constraints [A1] [A2] [A3] [A4]}

{wfx}
```

and the default file `g16.sub` is:

```
#!/bin/sh
#SBATCH --job-name={jobname}
#SBATCH --nodes=1
#SBATCH --ntasks={cpus}
#SBATCH --output={jobname}.logfile
#SBATCH --time={time}
{nodex #SBATCH --exclude=[nodenumbs]}

#Defining Gaussian Parameters
export g16root={soft_path}
source $g16root/bsd/g16.profile
export GAUSS_EXEDIR=$g16root
export LD_LIBRARY_PATH=$g16root
export GAUSS_SCRDIR=${SLUMR_SUBMIT_DIR}
cd ${SLURM_SUBMIT_DIR}

#Loading modules
module load intel/2023.2.1
export PATH=$g16root:$PATH
export LD_LIBRARY_PATH=$g16root
```

4.5.2 ORCA 4

The alias for the ORCA 4 package is orca, so the corresponding line in the parameters file should be:

```
software orca
```

In the `config.txt` there must be a line with the absolute path to the directory where the executable for orca is located:

```
path_orca=/home/pub/software/orca_4_1_2_linux_x86-64_openmpi313/
```

and a line to specify that the default alias for orca is simply orca:

```
default_orca=orca
```

Keywords for different type of calculations are specified in the `keywords.txt` file with the syntax `alias;calculation type;keywords`:

```
orca;opt;opt
orca;reopt;opt
orca;opt-ts;optTS Freq
orca;opt-ts-guided;optTS Freq
orca;reopt-ts;optTS Freq
orca;freq;freq
orca;cdft;NPA Hirshfeld
```

There also must be two files `orca.inp` and `orca.sub` in the `Models/` directory. The default file `orca.inp` is:

```
# ======================================================================
# Orca input file for FASTCAR
# ======================================================================
%pal nprocs {cpus} end
%maxcore 2000
! {keywords}  {method} {solvent} {dispersion}
! PrintBasis {basis-set}
{int coord %geom
    TS_Mode \{[T] [A1] [A2] [A3] [A4]\} end
end}
%output
     print[p_mos] 1
 end #output
* xyz  {charge}   {mult}
{geo}
*
```

and the default file `g16.sub` is:

```
#!/bin/sh
#SBATCH --job-name={jobname}
#SBATCH --nodes=1
#SBATCH --ntasks={cpus}
```

```
#SBATCH --output={jobname}.logfile
#SBATCH --time={time}
{nodex #SBATCH --exclude=[nodenumbs]}

# Loading modules
module load gnu12/12.3.0
module load openmpi4/4.1.6

export PATH={soft_path}:$PATH

#Starting calculation
time {soft_path}orca {jobname}.inp > {jobname}.out
```

4.6 How to add a new package

If one wants to make FASTCAR compatible with a new package, here are the steps to follow:

- Chose an alias for your package.
- Add a line in `config.txt` with the path to the directory of the package.
- Add a line in `config.txt` to specify the alias of the default version for this package.
- Create in the `Models/` directory two model files for input and submission script named with the chosen alias. Put the keywords listed in the right places, if some keywords are not useful they can be left out.
- Add keywords for each type of calculation you need in the `keywords.txt` file.

5 Identifying duplicates

FASTCAR is using SpyRMSD to identify duplicates in CREST results and in reoptimised structures to not waste calculation time and to give accurate results. This section aims to expose how the duplicate identification works.

SpyRMSD gives a measure of how similar to molecular structures are by calculating the symmetry corrected root mean square deviation of the two structures (for technical details, see the SpyRMSD documentation). The smaller the RMSD is, the more similar the two structures are (0 if the two structures are perfectly identical). To decide if two conformers are similar, users must set a threshold, if the RMSD for two conformers is below these threshold, they are considered identical. The default threshold is 0.1, but the value is dependant on the type of structures studied (the number of atoms, the flexibility of the system...) and on the goal of the user. Users are encouraged to test different threshold values to find the most suited one. Several tools are available in FASTCAR to help these tests.

As said, FASTCAR is identifying duplicates at two steps, in the CREST results and after reoptimisation. In general, one can say that the threshold used for the reoptimised structure can be lower than the one for the reoptimised structures.

6 Transition states screening

For transition states, FASTCAR check once the conformers are reoptimised that they correspond to the same reaction as the one of the starting transition state. There are currently two ways of doing that can be selected in the parameters file.

6.1 Scalar product

This solution can be choosen with the option `scalprod` for the `TS screening` keyword, it's the default one. In this case, FASTCAR will compute the scalar product of the two normal modes associated with the imaginary frequency and if the result is higher than a threshold the two transition states will be considered corresponding to the same reaction. To do that, the conformer is first aligned with the starting structure with the function `spatial.transform.Rotation.align_vectors` from the `scipy` library. Then vectors containing the 3N components of the normal modes are created for the conformer and the starting structure and the scalar product is computed. This is done for the original geometry of the conformer and the aligned one (since sometimes, when the conformer is too different, the alignment actually make things worse), only the highest value of the scalar product is considered. The default value for the threshold is fairly low at 0.3 due to the possibility of having very different structures for the conformers. The results of all scalar product (aligned and not aligned) are written in files named compTS[_n].log (n being the number of the loop), users are encouraged to go see the values they get for their system and adjust the threshold if needed.

6.2 Active atoms

If the scalar product is not relevant for a system (for instance if the conformers have geometries too different from the geometry of the starting structure), the active atoms method can be chosen with the option `activats` for the `TS screening` keyword. With this method, the user need to specify which atoms are moving the most during the reaction, this is done in the parameters file with the keyword `active atoms` followed by the numbers of the atoms. Then for each conformer, FASTCAR will check if these atoms are in the n atoms with the highest displacement vector in the normal mode associated with the imaginary frequency (n equals the number of active atoms given + 3). If not, the conformer will be flagged as garbage, *i.e.* not corresponding to the reaction of interest.