Installation

Make sure you have Python 3.7 or newer (`python --version`).

If you can't call the qforce executable afterwards, make sure you have the python bin in your PATH.

With pip

To install Q-Force with `pip`:

```
pip install qforce
```

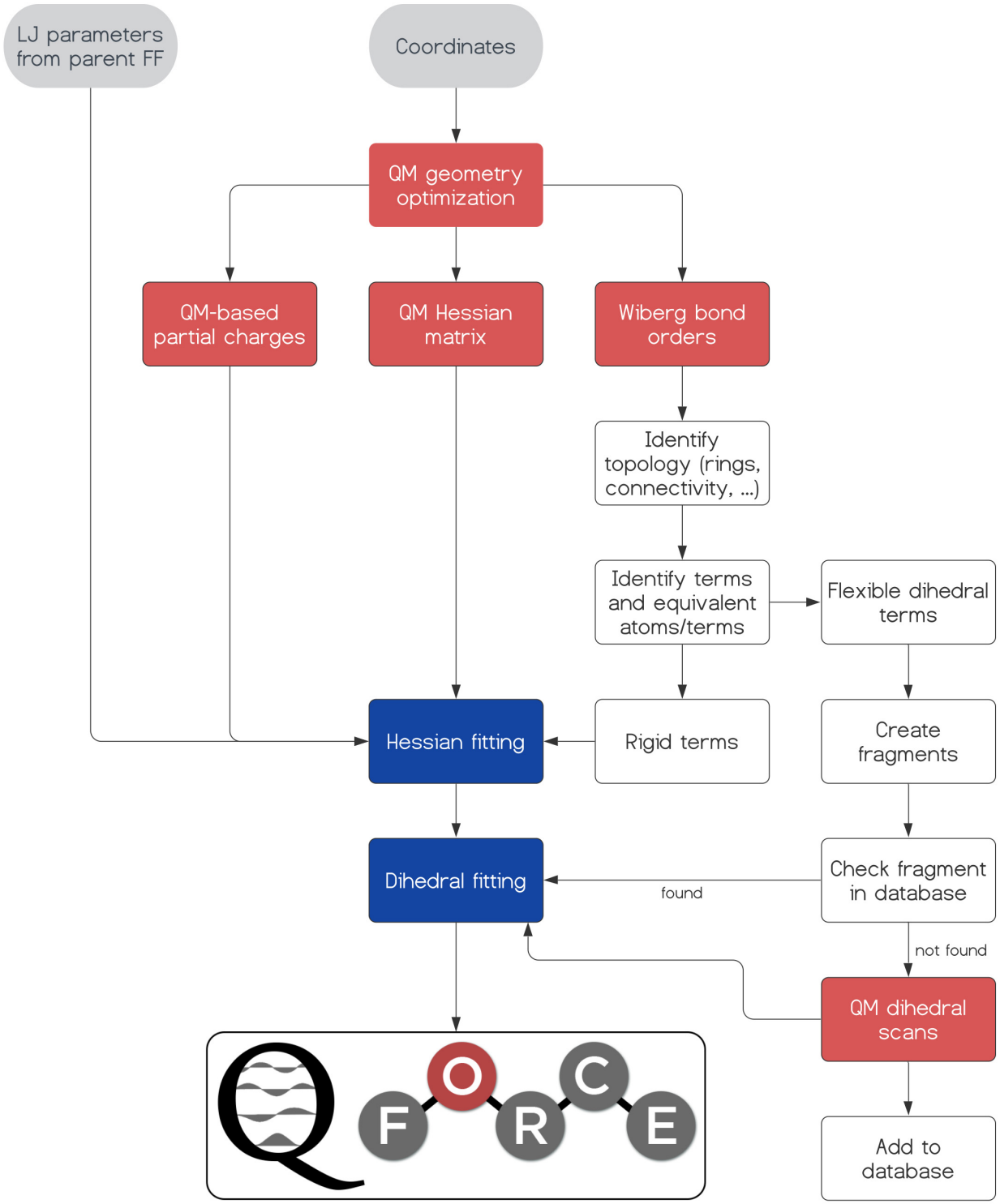
If you work in a shared environment, add `--user`.

From GitHub

To install Q-Force from GitHub:

```
git clone https://github.com/selimsami/qforce.git
cd qforce
python setup.py install
```

Method



Q-Force flowchart. Gray: input, red: QM calculations, blue: fitting

For the detailed methodology, please check the corresponding manuscript:

Sami, S.; Menger, M. F. S. J.; Faraji, S.; Broer, R.; Havenith, R. W. A., Q-Force: Quantum

Mechanically Augmented Molecular Force Fields. Journal of Chemical Theory and Computation 2021, 17 (8), 4946-4960.

Usage

Q-Force is run in multiple stages. These stages are explained below. At each stage, an options file can be provided to change the default settings with `-o file_name`. Possible options are listed in Options.

```
      ____          ______
     / __ \        |  ____|
    | |  | |______| |__ ___  _ __ ___ ___
    | |  | |______|  __/ _ \| '__/ __/ _ \
    | |__| |      | | | (_) | | | (_|  __/
     \___\_\      |_|  \___/|_|  \___\___|

                Selim Sami
          University of Groningen - 2020
          ==============================


usage: qforce [-h] [-o options] file


  positional arguments:
  -----------------------------------------------------
    file                      Input coordinate file mol.ext (ext: pdb, xyz, gro, ...)
                              or directory (mol or mol_qforce) name.


  optional arguments:
  -----------------------------------------------------
    -h/--help                 show this help message and exit
    -o/--options              File name for the optional options.
```

1) Creating the initial QM input

et's assume that we have a coordinate file called **mol.ext** for a molecule named **mol**. The extension (**ext**) can be anything that is supported by ASE (xyz, pdb, gro, ...). reate the initial QM input (choosing the QM Software is described in Options) by running the following command `qforce mol.ext`

This creates a directory called *mol_qforce*. In it, you can find **mol_hessian.inp**. Run this calculation on a cluster or locally, and place the output(s) in the same directory.

2) Treating the flexible dihedrals

If your molecule contains flexible dihedrals and if the treatment of flexible dihedrals are not turned off, then fragments and the corresponding QM inputs are created for all unique flexible dihedrals inside the subdirectory *fragments* with:

`qforce mol` (or `qforce mol_qforce`, or `qforce mol.ext`)

Run these calculations on a cluster or locally, and place the output in the same subdirectory.

3) Creating the force field

Now that all necessary QM results are available, the fitting of the force field is done with:

`qforce mol` (or `qforce mol_qforce`, or `qforce mol.ext`)

4) Output

Done! Q-Force generates several outputs:

- Force field files in GROMACS format (.gro, .itp, .top)
- Force field validation:
  - QM vs MM vibrational frequencies (frequencies.txt, frequencies.pdf)
  - QM vs MM dihedral profile(s) in the *fragments* subdirectory (.pdf)
- MM vibrational modes (frequencies.nmd) that can be visualized in VMD

Examples

Here are two examples of how Q-Force can be used: In default settings and with some customization. For the purposes of these examples, whenever you need an additional file, QM outputs or otherwise, they are provided in the directory *necessary_files*.

First, please get the example files by:

```
git clone https://github.com/selimsami/qforce_examples.git
```

Default settings

Creating the initial QM input

Find in *examples/gaussian/default_settings* a coordinate file (propane.xyz) for the propane molecule.

Let's first create the QM input file:

```
qforce propane.xyz
```

This will create a *propane_qforce* directory, and in it, you will find 'propane_hessian.inp'. Now run this QM calculation and put the necessary output files (.out, .fchk) in the same directory. (remember: the output files are available in *necessary_files*)

Treating the flexible dihedrals

Now we can run Q-Force again from the same directory to create fragments and the corresponding QM dihedral scan input files by:

```
qforce propane
```

This will create all the necessary input files in the subdirectory *propane_qforce/fragments*. Then, run these calculations and put the output file(s) (.out) in the same subdirectory.

Creating the force field

Now that all necessary QM data is available, let's create our force field:

```
qforce propane
```

You can now find the Q-Force force field files in the *propane_qforce* directory.

Custom settings

Find in *examples/gaussian/custom_settings* a coordinate file (benzene.pdb) for the benzene molecule. In this example, we look at some of the custom settings available with Q-Force and how they can be executed. The custom settings are provided with an external file with:

```
qforce benzene.pdb -o settings
```

Now let's create the **settings** file.

Custom Lennard-Jones interactions

By default, Q-Force determines the atom types for Lennard-Jones interactions automatically. Alternatively, the user can also provide atom types manually, for a force field of their choice. Here, we choose to use the GAFF force field by adding the following line to the **settings** file:

```
[ff]
lennard_jones = gaff
```

With this command, the user also has to provide the atom types manually in the 'benzene_qforce' directory in a file called "ext_lj". In this file, every line should contain the atom type of one atom in the same order as the coordinate file.

Conversion to job script

Often the QM calculations are needed to be submitted as jobs in supercomputers. For large molecules Q-Force can have a large number of QM dihedral scans that needs to be performed and therefore it may be convenient to have input files converted to job scripts. This can be done by adding the **[qm::job_script]** block to the **settings** file:

```
[qm::job_script]
#!/bin/bash
#SBATCH --time=1-00:00:00
#SBATCH -o <jobname>.out

g16<<EOF
<input>
EOF
```

Here we make a SLURM job script. Two placeholders that can be used are **<outfile>** and **<input>**. **<jobname>** gets replaced by the name of the calculation, for example in the case of the 'benzene_hessian.inp', it will be 'benzene_hessian.out'. **<input>** is where the content of the QM input file will be placed.

Creating the initial QM input

Now that we know what these settings do, let's supply them to Q-Force:

```
qforce benzene.pdb -o settings
```

Again, this will create a *benzene_qforce* directory, and in it, you will find 'benzene_hessian.inp', this time as a job script instead of an input file. Now run this QM calculation and put the output file (.out) and the formatted checkpoint file (.fchk) in the same directory.

Creating the force field

As benzene does not have any flexible dihedrals, the second step is skipped in this case. Make sure you have also added this time the **ext_lj** file in *benzene_qforce* and then we can already create the force field with:

```
qforce benzene -o settings
```

You can now find the necessary force field files in the *benzene_qforce* directory. As you will notice, in this case GAFF atom types are used.

Choosing the QM software

The default QM software is Gaussian. If the user wants to use another QM software (current alternative: Q-Chem), this can be indicated in the same **settings** file:

```
[qm]
software = qchem
```

An example for running Q-Force with Q-Chem can be found in the *examples/qchem /default_settings* directory. This works in the same way as the first example, except the additional argument for choosing the QM software, as shown above.

Options

[ff]

n_equiv, **int:**

```
default: 4
Number of n equivalent neighbors needed to consider two atoms equivalent
Negative values turns off equivalence, 0 makes same elements equivalent
```

n_excl, **int:**

```
default: 2, from Choices(2, 3)
Number of first n neighbors to exclude in the forcefield
```

lennard_jones, **str:**

```
default: opls_auto, from Choices(gromos_auto, gromos, opls_auto, opls, gaff, gaff2, charm
Lennard jones method for the forcefield
```

ext_charges, **bool:**

```
default: False, from Choices(True, False)
Use externally provided point charges in the file "ext_q" in the job directyory
```

charge_scaling, **float:**

```
default: 1.2
Scale QM charges to account for condensed phase polarization (should be set to 1 for gas
```

use_ext_charges_for_frags, **bool:**

```
default: False, from Choices(True, False)
If user chooses ext_charges=True, by default fragments will still use the chosen QM metho
determining fragment charges. This is to avoid spuriously high charges on capping hydroge
However, using QM charges for fragments and ext_charges for the molecule can also result
inaccuracies if these two charges are very different.
```

ff::exclusions, **LiteralBlock**

```
Additional exclusions (GROMACS format)
```

ff::pairs, **LiteralBlock**

```
Switch standard non-bonded interactions between two atoms to pair interactions
(provide atom pairs in each row)
```

ext_lj_lib, **folder:**

```
Path for the external FF library for Lennard-Jones parameters (GROMACS format).
```

ext_lj_fudge, **float:**

```
Lennard-Jones fudge parameter for 1-4 interactions for when "lennard_jones" is set to "ex
```

ext_q_fudge, **float:**

```
Coulomb fudge parameter for 1-4 interactions for when "lennard_jones" is set to "ext".
```

ext_comb_rule, **int:**

```
Choices(1, 2, 3)
Lennard-Jones combinations rules for when "lennard_jones" is set to "ext" (GROMACS number
```

ext_h_cap, **str:**

```
Name of the atom type for capping hydrogens for when "lennard_jones" is set to "ext"
```

all_rigid, **bool:**

```
default: False, from Choices(True, False)
Set all dihedrals as rigid (no dihedral scans)
```

res_name, **str:**

```
default: MOL
Residue name printed on the force field file (Max 5 characters)
```

ff::_polar_not_scale_c6, **LiteralBlock**

```
Specifically not scale some of the atoms
```

[qm]

software, **str:**

```
default: gaussian, from Choices(gaussian, qchem, orca, xtb)
QM software to use
```

qm::job_script, **LiteralBlock**

```
To turn the QM input files into job scripts
```

scan_step_size, **float:**

```
default: 15.0
Step size for the dihedral scan (360 should be divisible by this number ideally)
```

charge, **int:**

```
default: 0
Total charge of the system
```

multiplicity, **int:**

```
default: 1
Multiplicity of the system
```

memory, **int:**

```
default: 4000
Allocated memory for the QM calculation (in MB)
```

n_proc, **int:**

```
default: 1
Number of processors to set for the QM calculation
```

vib_scaling, **float:**

```
default: 1.0
Scaling of the vibrational frequency for the corresponding QM method (not implemented)
```

dihedral_scanner, **str:**

```
default: relaxed_scan, from Choices(relaxed_scan, xtb-torsiondrive)
Use the internal relaxed scan method of the QM software or the Torsiondrive method using
```

[qm::software(gaussian)]

charge_method, **str:**

```
default: cm5, from Choices(cm5, esp)
```

method, **str:**

```
default: PBEPBE
QM method to be used
```

dispersion, **str:**

```
default: EmpiricalDispersion=GD3BJ
Dispersion method - leave it empty to turn off
```

basis, **str:**

```
default: 6-31+G(D)
QM basis set to be used - leave it empty to turn off
```

solvent_method, **str:**

```
Include implicit solvent for the complete parametrization
```

[qm::software(orca)]

charge_method, **str:**

```
default: esp, from Choices(cm5, esp)
```

qm_method_opt, **str:**

```
default: r2SCAN-3c
QM method to be used for geometry optimisation
```

qm_method_hessian, **str:**

```
default: B3LYP D4 def2-TZVP def2/J RIJCOSX
QM method to be used for hessian calculation
Note: The accuracy of this method determines the accuracy of bond,
angle and improper dihedral.
```

qm_method_charge, **str:**

```
default: HF 6-31G*
QM method to be used for charge derivation
Note: Method chosen according to the standard RESP procedure.
```

qm_method_sp, **str:**

```
default: PWPB95 D4 def2-TZVPP def2/J def2-TZVPP/C notrah RIJCOSX tightSCF
QM method to be used for dihedral scan energy calculation.
Note: The accuracy of this method determines the accuracy of
flexible dihedral.
```

[qm::software(qchem)]

charge_method, **str:**

```
default: cm5, from Choices(cm5, resp)
```

method, **str:**

```
default: PBE
QM method to be used
```

dispersion, **str:**

```
default: d3_bj, from Choices(d3, d3_bj, d3_bjm, d3_zero, d3_op, empirical_grimme)
Dispersion (enter "no"/"false" to turn off)
```

basis, **str:**

```
default: 6-31+G(D)
QM basis set to be used (enter "no"/"false" to turn off)
```

max_scf_cycles, **int:**

```
default: 100
Number of maximum SCF cycles
```

max_opt_cycles, **int:**

```
default: 100
Number of maximum optimization cycles
```

xc_grid, **int:**

```
default: 3, from Choices(0, 1, 2, 3)
DFT Quadrature grid size
```

cis_n_roots, **int:**

```
Number of CIS roots to ask
```

cis_singlets, **bool:**

```
Choices(True, False)
CIS singlets turned on or off
```

cis_triplets, **bool:**

```
Choices(True, False)
CIS triplets turned on or off
```

cis_state_deriv, **int:**

```
Sets CIS state for excited state optimizations and vibrational analysis
```

solvent_method, **str:**

```
Include implicit solvent for the complete parametrization
```

[qm::software(xtb)]

charge_method, **str:**

```
default: xtb
xTB only allows Mulliken charge.
```

xtb_command, **str:**

```
default: --gfn 2
Extra command line passed to the xtb executable
```

[scan]

do_scan, **bool:**

```
default: True, from Choices(True, False)
Perform dihedral scan for flexible dihedrals
```

avail_only, **bool:**

```
default: False, from Choices(True, False)
Skip dihedrals with missing scan data and accept scan data that is missing data points.
False: Exit if scan data for a dihedral is missing or if it has missing data points)
```

frag_threshold, **int:**

```
default: 3
Number of neighbors after bonds can be fragmented (0 or smaller means no fragmentation)
```

break_co_bond, **bool:**

```
default: False, from Choices(True, False)
Break C-O type of bonds while creating fragments (O-C is never broken)
```

method, **str:**

```
default: qforce, from Choices(qforce, gromacs)
Method for doing the MM relaxed dihedral scan
```

gromacs_exec, **str:**

```
default: gmx
The executable for gromacs - necessary if scan method is gromacs
```

n_dihed_scans, **int:**

```
default: 5
Number of iterations of dihedral fitting
```

scan::symmetrize, **LiteralBlock**

```
Symmetrize the dihedral profile of a specific dihedral by inputting the range
For symmetrizing the dihedral profile between atoms 77 and 80 where 0-180 is inversely
equivalent to 180-360:
77 80 = 0 180 360 : +-
```

plot_fit, **bool:**

```
default: False, from Choices(True, False)
Save extra plots with fitting data
```

frag_lib, **folder:**

```
default: /home/docs/qforce_fragments
Directory where the fragments are saved
```

batch_run, **bool:**

```
default: False, from Choices(True, False)
Skip the dihedrals that are generated but not computed
```

[terms]

urey, **bool:**

```
default: True, from Choices(True, False)
Turn urey FF term on or off
```

dihedral/rigid, **bool:**

```
default: True, from Choices(True, False)
Turn dihedral/rigid FF term on or off
```

dihedral/improper, **bool:**

```
default: True, from Choices(True, False)
Turn dihedral/improper FF term on or off
```

dihedral/flexible, **bool:**

```
default: True, from Choices(True, False)
Turn dihedral/flexible FF term on or off
```

dihedral/inversion, **bool:**

```
default: True, from Choices(True, False)
Turn dihedral/inversion FF term on or off
```

non_bonded, **bool:**

```
default: True, from Choices(True, False)
Turn non_bonded FF term on or off
```