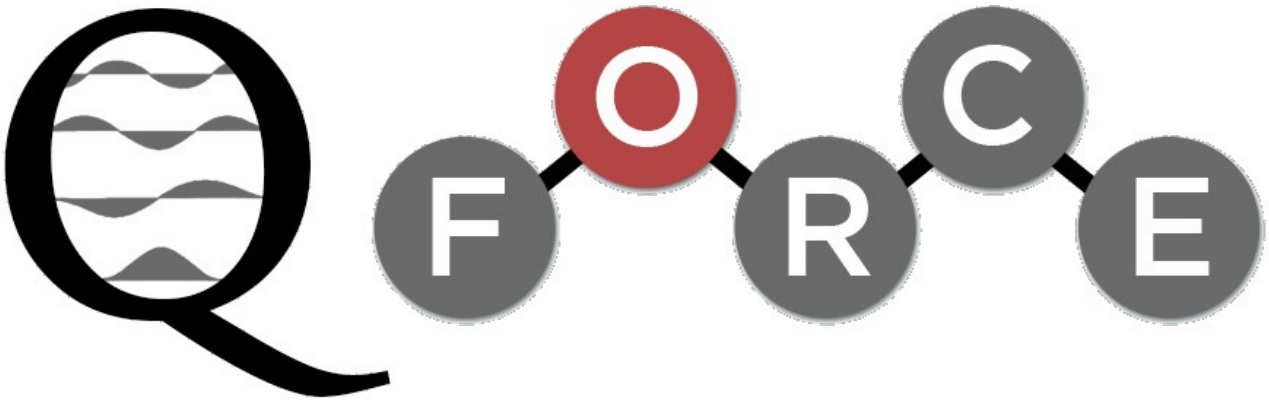


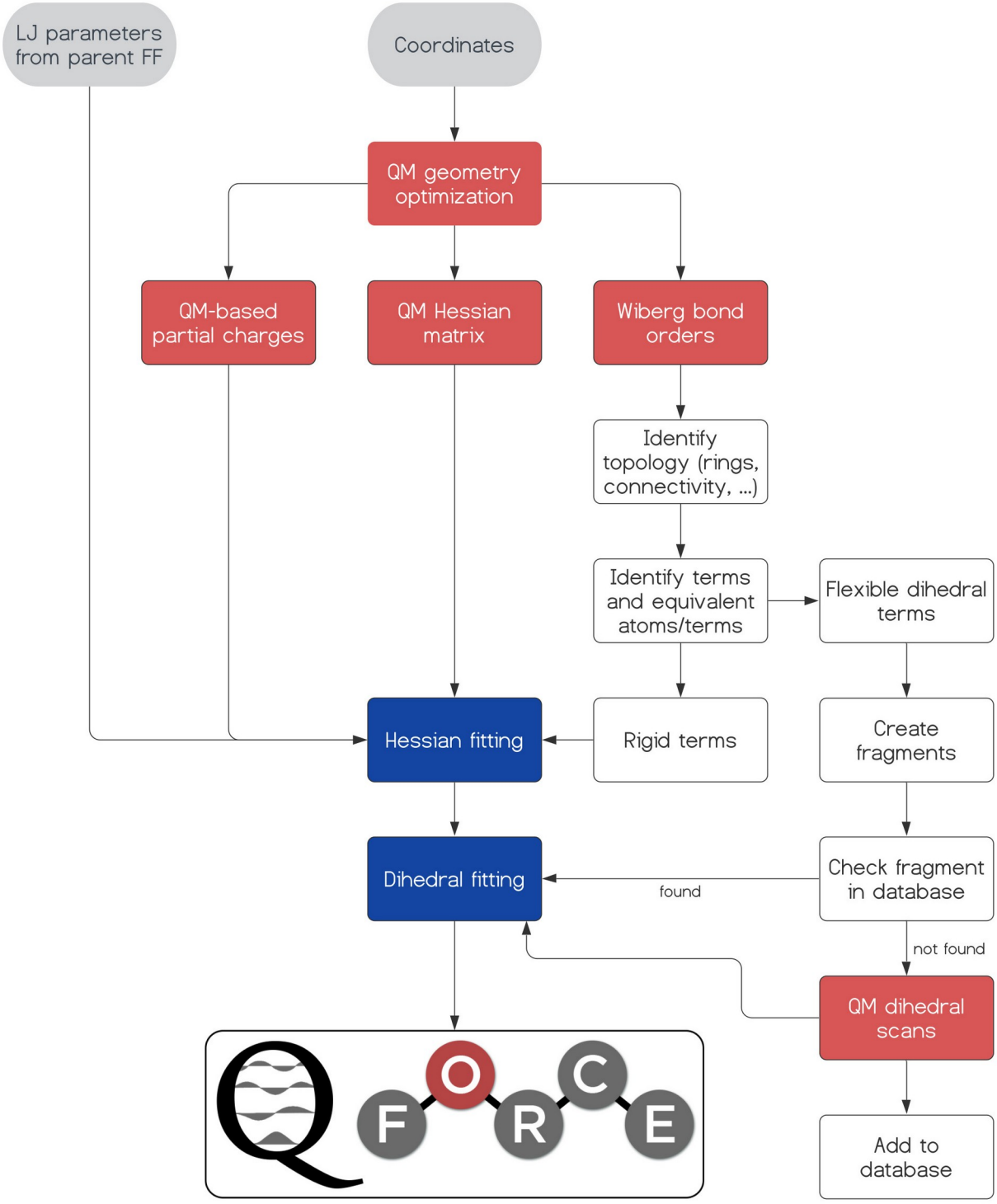
Quantum Mechanically augmented molecular force fields

Please see the Documentation.

If you use Q-Force, please cite:

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.

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 ChemicalTheory and Computation* 2021, 17 (8), 4946-4960. [link]

Installation

Make sure you have Python 3.7 or newer.

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
```

From Binary:

To install Q-Force from binary form:

```
cd $HOME_SOLVATE/nodes/qforce && ./qforce.install
```

From Source:

To compile Q-Force from source form:

1. Install QFORCE and PYINSTALLER from the pip repository:

```
pip install qforce
pip install pyinstaller
```

2. Go to the QFORCE installation folder (/home/$USER/.local) and create the "hook-ase" and "hook-sip" folders.

ASE and SIP are libraries that are not found by default by the PYINSTALLER packager. This information was obtained by analyzing line by line of the debug. Inside these folders are simple Python scripts.

Script 1 hook-ase.py:
mkdir hook-ase
with hook-ase.py script containing:
```
from PyInstaller.utils.hooks import collect_all
datas, binaries, hiddenimports = collect_all('ase')
```

Script 2 hook-sip.py:
mkdir hook-sip
with hook-sip.py script containing:
```
from PyInstaller.utils.hooks import collect_all
datas, binaries, hiddenimports = collect_all('sip')
```
---

3. Execute the command:

```
pyinstaller --onefile
--additional-hooks-dir=/PATH/TO/DIR/hook-ase
--additional-hooks-dir=/PATH/TO/DIR/hook-sip
--add-data 'PATH/TO/LIBRARY/DIRECTORY/pulp:pulp'
--add-data '/PATH/TO/DIR/qforce:qforce'
--add-data '/PATH/TO/DIR/qforce/qforce/qm:qforce/qm'
--add-data '/PATH/TO/DIR/qforce/qforce/data:qforce/data'
--add-data '/PATH/TO/DIR/qforce/qforce/tests:qforce/tests'
--add-data '/PATH/TO/DIR/qforce/qforce/molecule:qforce/molecule'
PATH/TO/BINARY/qforce
```

or simply:

```
pyinstaller qforce.spec
```

Explanation: The --onefile command indicates that a single executable will be created, --additional-hooks-dir indicates that the library is being included manually, and --add-data indicates the inclusion of files that QFORCE needs to function.

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

Let'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, ...). Create 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.

Usage

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 the input file '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 procedure will create all the necessary input files in the subdirectory named *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.

Examples

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 -o <jobname>.out
g16<<EOF
<input>
EOF
```

Here we make a SLURM job script. Two placeholders that can be used are *<input>* and *<outfile>*. *<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.

Examples

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 the input "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.pdb -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 alternatives: *Q-Chem*, *ORCA*, and *xTB*), 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, charmm36, ext)
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 phase).

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 method for determining fragment charges. This is to avoid spuriously high charges on capping hydrogens. However, using QM charges for fragments and ext_charges for the molecule can also result in 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).

Options

[ff]

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 “ext”.

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 numbering).

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.

Options

[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 *xTB*.

Options

[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.

Options

[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.

Options

[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 optimization.

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.

Options

[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.

License

APACHE LICENSE
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works

thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License.

Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License.

Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution.

You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

- You must give any other recipients of the Work or Derivative Works a copy of this License; and
- You must cause any modified files to carry prominent notices stating that You changed the files; and
- You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

- If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions.

Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks.

This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty.

Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability.

In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability.

While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS