MANUAL

QuickFFmn 2016

by
Shaohong L. Li, and Donald G. Truhlar
Department of Chemistry and Supercomputing Institute,
University of Minnesota, Minneapolis, MN 55455-0431, USA

QuickFFmn is an extension of *QuickFF1.0.1*
by
Louis Vanduyfhuys, Steven Vandenbrande, Toon Verstraelen,
R. Schmid, M. Waroquier, V. Van Speybroeck
Center for Molecular Modeling, Ghent University,
Technologiepark 903, 9052 Zwijnaarde, Belgium

Program version date: June 9, 2016
Documentation version date: July 21, 2021

Table of Contents

1. Licensing

QuickFFmn - version 2016 is licensed under the Apache License, Version 2.0.
The manual of *QuickFFmn* - version 2016 is licensed under CC-BY-4.0.
Publications of results obtained with the *QuickFFmn* - version 2016 software should cite the program and/or the article describing the program.

No guarantee is made that this software is bug-free or suitable for specific applications, and no liability is accepted for any limitations in the mathematical methods and algorithms used within. No consulting or maintenance services are guaranteed or implied.

The use of the *QuickFFmn* - version 2016 implies acceptance of the terms of the licenses.

2. Introduction

QuickFF (http://molmod.github.io/QuickFF) is a Python package for deriving force fields from ab initio input data. This version of QuickFF, called QuickFFmn 2016, is developed at the University of Minnesota based on QuickFF1.0.1 developed at the Ghent University, Belgium, with extra capabilities implemented.

3. Citation

S. L. Li and D. G. Truhlar, QuickFFmn 2016 (http://comp.chem.umn.edu/quickffmn/) based on QuickFF – version 1.0.1 (http://molmod.github.io/QuickFF) as described in L. Vanduyfhuys, S. Vandenbrande, T. Verstraelen, R. Schmid, M. Waroquier, and V. Van Speybroeck, J. Comput. Chem. **36**, 1015 (2015) (http://dx.doi.org/10.1002/jcc.23877).

4. Extra capabilities added in QuickFFmn 2016

- Simons-Parr-Finlan (SPF)[1] potential for bond stretches,

  $$U(R) = \frac{1}{2}k\left(\frac{R - R_e}{R}\right)^2$$

  where U is potential energy, R is a bond length, k and R_e are parameters.
- Harmonic-cosine potential[2] for valence bends,

  $$U(\theta) = \frac{1}{2}k(\cos\theta - \cos\theta_e)^2$$

  where U is potential energy, θ is a bond angle, k and θ_e are parameters.

5. Additional changes in QuickFFmn 2016

- Sulfur is deemed an "important" atom besides C, N, O for estimating atom types at 'high' level.
- Manual definition of atom connectivity can be passed via quickff.System.from_files function.

6. Installation

Install the original QuickFF1.0.1 and its dependencies (see http://molmod.github.io/QuickFF; one copy of QuickFF1.0.1 is provided), then replace the following files with the provided ones in the src_qffmn directory:

- fftable.py
- model.py
- perturbation.py
- program.py
- system.py
- terms.py
- tools.py

7. Using the extra capabilities

To use SPF and/or harmonic-cosine potential, add the following parameter(s) when calling system.determine_ics_from_topology (see sample code in Section 9):

```
stretch_pot_kind='spf'
bend_pot_kind='harmcos'
```

8. Test sets

The following test sets are provided in the testset/ directory. (See readme.txt in the test sets for more details.)

- water_harmonic: constructing valence force field (bond stretches + bending) for water using original harmonic terms.
- water_new: constructing valence force field (bond stretches + bending) for water using newly implemented SPF and harmonic-cosine terms.

9. Sample Python code for constructing FF potential

```
from quickff import *

#--- Defining the system ---

#Read system from input files
system = System.from_files(['gaussian.fchk'])

#Guess atom types
system.guess_ffatypes('high')

#Determine internal coordinates from topology
system.determine_ics_from_topology(stretch_pot_kind='spf',
bend_pot_kind='harmcos')

#--- Defining the model and program ---

model = Model.from_system(system, ai_project=True)
program = Program(system, model)

#--- Constructing the force field ---

#Estimate rest angle and multiplicity of dihedral potentials from geometry
model.val.determine_dihedral_potentials(system)

#Determine the coordinates of the perturbation trajectories
trajectories = program.generate_trajectories()

#Estimate all pars for bonds, bends and opdists
fftab = program.estimate_from_pt(trajectories)

#Refine force constants using a Hessian LSQ cost
fftab = program.refine_cost()

#--- Generating output ---
```

```
fftab.print_screen()
fftab.dump_ffit2('pars_ffit2.txt')
fftab.dump_yaff('pars_yaff.txt')
```

10. Modifications to the code

Search for #SHLL for modifications and comments in the following files:

- fftable.py
- model.py
- perturbation.py
- program.py
- system.py
- terms.py
- tools.py

11. Additional references

[1] G. Simons, R. G. Parr, and J. M. Finlan, J. Chem. Phys. **59**, 3229-3934 (1973).

[2] K. R. Yang, X. Xu, and D. G. Truhlar, J. Chem. Theory Comput. **10**, 924-933 (2014).