The Journal of Open Source Software

qtools: A Python toolset for the Q molecular simulation package.

Miha Purg[1]

1 Department of Chemistry-BMC, Uppsala University, Box 576, SE-751 23 Uppsala, Sweden

DOI: 10.21105/joss.00930

Software

- Review
- Repository
- Archive

Submitted: 05 September 2018
Published: 06 September 2018

Introduction

Molecular modeling and simulations are becoming an increasingly integral part of enzymology, complementing experimental measurements with a direct view into the dynamics and reactivity of enzymes at the atomic level. The multiscale quantum mechanics/molecular mechanics (QM/MM) methodology has proven to be particularly valuable for studying enzyme-catalyzed reactions, gaining widespread recognition evidenced by the Nobel prize in Chemistry in 2013. By providing a map of the energy landscape for (bio)chemical systems, these methods help to elucidate chemical mechanisms, rationalize substrate specificity, and estimate energetic contributions to catalysis, all of which might otherwise be difficult to quantify. Such detailed understanding of the forces and interactions governing the activity of enzymes is also tremendously valuable in the rational design of inhibitors as potential drug candidates. Furthermore, it can aid in the design of enzyme functionality, *e.g.* in search of environmentally friendly industrial biocatalysts.

One of the well-established methodologies for studying enzyme reactions is the empirical valence bond (EVB) method. (Warshel and Weiss 1980) In EVB, a chemical reaction is described within a valence bond formalism, representing individual valence states (*e.g.* reactants and products) with classical force-field potentials. Employing a classical model dramatically reduces the computational cost compared to traditional QM/MM methods allowing for substantially longer molecular dynamics simulations, although at the cost of accuracy. A particularly useful application of EVB is in the rapid screening of point-mutations, as demonstrated in our recent work. (Amrein et al. 2017) The EVB methodology is implemented in Q, a software package designed specifically for running molecular dynamics simulations in the context of free energy perturbation (FEP) and EVB. (Marelius et al. 1998),(Bauer et al. 2018) Due to its sleek and minimalistic design, the users would benefit greatly from a well-designed high-level interface, increasing the productivity and minimizing the use of system-specific, poorly tested, closed-source scripts.

Overview

`qtools` is a toolset comprised of a Python library `Qpyl` and a set of command-line tools, for use with the molecular simulation package Q. The motivation behind `qtools` is to provide users of Q with a structured, well-tested, open-source Python library for efficient setup, troubleshooting, calibration, and analysis of EVB simulations. In addition, a set of Unix command-line tools, which directly calls the `Qpyl` API, is bundled in the repository, providing a user interface which is particularly practical for use in typical high-performance-computing (HPC) situations. While EVB simulations are the main

focus of `qtools`, non-reactive FEP calculations (*e.g.* ligand binding and solvation free energies) are fully supported.

The functionality implemented in `Qpyl` is application-specific but relatively broad, encompassing several modules for wrapping Q executables, generating and parsing I/O files, generating, parsing and converting parameter files, analysing FEP, EVB, QCP free energy results, calibrating the EVB potential, and other. Most importantly, it exposes this functionality *via* a Python API, providing users with programmable means for preparing, simulating, and analysing EVB simulations. In addition to the EVB-related functionality, `qtools` implements the linear response approximation (LRA) method for estimating free energies. (Lee et al. 1992) The LRA method defines the free energy between two states, A and B, as:

$$\Delta G(A \rightarrow B) = \frac{1}{2}\Big[\langle U_B - U_A \rangle_A + \ \langle U_B - U_A \rangle_B\Big]$$

In `Qpyl`, LRA is used as an efficient and convenient post-processing approach for calculating the *virtual deletion* of amino-acid residues, providing estimates of their energetic contributions to catalysis. Additionally, `qpyl` allows the estimation of contributions to *reorganization energy*, (Muegge et al. 1997) which were suggested to be especially useful in computational design of enzymes. (Fuxreiter and Mones 2014)

Scientific publications employing `qtools` include (Amrein et al. 2017),(Purg, Elias, and Kamerlin 2017),(Bauer et al. 2018). A recently published chapter in Methods in Enzymology provides step-by-step instructions on using `qtools` in practice, from obtaining force-field parameters and input generation, to analysis and troubleshooting. (Purg and Kamerlin 2018)

Technical details

The main design characteristics of `qtools` are:

1. *Minimum dependencies.* The only dependency is Python 2.7, with the exception of one (non-essential) tool which requires `matplotlib`.
2. *Logic separation and extensibility.* The functionality is implemented in the Python library `Qpyl` and exposed *via* an API. A higher-level application/interface (GUI, web-service, *etc.*) should avoid implementing Q-related functionality and instead call the API, analogous to the bundled command-line tools.
3. *Thorough testing.* Continuous integration using `py.test` for Qpyl, and Bash regression tests for command-line tools.
4. *Strict sanity checks.* Invalid Q inputs, parameters and/or settings, charge imbalances, duplicate parameters, *etc.*, raise exceptions or propagate logging warnings.
5. *Balanced abstraction.* `qtools` tries to maintain a good balance between the level of user-control over Q (*i.e.* relatively low abstraction level) and ease-of-use.

Python library Qpyl

Core functionality implemented in `Qpyl.core` includes generator and parser objects for Qfep, Qdyn, and Qcalc input and output files, as well as for Q library, parameter, and coordinate files. Several external library and parameter files are supported, including AMBER (Maier et al. 2015) `.parm`, `.frcmod`, `.lib` and `.prepin` files, and output from MacroModel. (Release 2016) `Qpyl.core.qtopology` implements an internal topology builder for mapping the system's bonding patterns with force-field parameters. In conjunction with the force-field potential energy functions implemented in `Qpyl.core.qpotential` one can

use the topology object for direct evaluation of individual topological components of the system.

Example 1: Joining parameter files

```
>>> from Qpyl.core.qparameter import QPrm
>>> qprm = QPrm("oplsaa")
>>> qprm.read_prm("qoplsaa.prm")
>>> qprm.read_prm("ligand.prm") # raises QPrmError on parameter conflicts
>>> open("ooplsaa_ligand.prm", "w").write(qprm.get_string())
```

Higher-level functionality includes wrapper objects for Q programs, allowing for a thread-enabled automatic generation of inputs, system calls to Q executables, and analysis of outputs. The analysis objects in `Qpyl.qanalysis` extract/calculate relevant free energies (activation, reaction and FEP free energies) and perform basic statistical analysis of the simulations. Quantum classical path (QCP) and group exclusion calculations implemented in the recent version of Q are fully supported. In addition, `Qpyl` implements the automated calibration of (two-state) EVB potentials, the calculation of *group contributions* using the LRA method, and provides convenient methods for generating Qdyn input files for equilibration and FEP simulation stages. `Qpyl.qmakefep` implements a semi-automated approach for determining the changes in force-field parameters between (valence) states, as defined in a Q FEP file. The Q FEP file is built automatically from the provided parameter files (for each state), the initial state coordinate file, and a user-defined mapping of atoms in each state. It further provides sanity checks for missing atoms and parameters, charge imbalances, *etc.*

Example 2: Analysing FEP outputs

```
>>> from Qpyl.qanalysis import QAnalyseFeps
>>> qaf = QAnalyseFeps(["1/qfep.out", "2/qfep.out"])
>>> print qaf.dgas.values()  # activation free energies
[12.21, 12.55]
```

Example 3: Generating a FEP file

```
>>> from Qpyl.qmakefep import make_fep
>>> fepstring = make_fep(qmap_file="atom_mapping.qmap", pdb_file="sn2_start.pdb",
                         parm_files=["cl.prm", "mebr.prm", "mecl.prm", "br.prm"],
                         lib_files=["cl.lib", "mebr.lib", "mecl.prm", "br.lib"])
>>> open("sn2.fep", "w").write(fepstring)
```

Command-line interface

`qtools` comes with a command-line interface allowing Q-users direct access to the functionality implemented in `Qpyl`. The CLI tools are designed for use in a Unix environment, are namespaced by using a prefix 'q_' (*e.g.* `q_analysefeps.py`), are non-interactive and rely extensively on command-line arguments. Usages include: parameter conversion from MacroModel (OPLS) or AMBER formats (`q_ffld2q.py`, `q_amber2q.py`), FEP file generation (`q_makefep.py`), input generation (`q_genrelax.py`, `q_genfeps.py`), Qfep wrappers (`q_mapper.py`, `q_automapper.py`), Qcalc wrapper (`q_calc.py`), and analysis of simulations (`q_analysefeps.py`, `q_analysedyns.py`).

Graphical interface

Finally, `qtools` comes with a GUI application `q_plot.py` for visualizing results and troubleshooting simulations. `q_plot.py` displays the data generated by CLI tools, *i.e.* free energy profiles, LRA contributions, LRA group contributions, potential energy breakdown, kinetic energy, temperature, *etc.*, which is especially useful in troubleshooting simulations in Q. Some examples of use are demonstrated in reference (Purg and Kamerlin 2018)

Acknowledgments

I thank Dr. Janez Mavri and Dr. Paul Bauer for all the helpful discussions and support.

Amrein, Beat Anton, Fabian Steffen-Munsberg, Ireneusz Szeler, Miha Purg, Yashraj Kulkarni, and Shina Caroline Lynn Kamerlin. 2017. "CADEE: Computer-Aided Directed Evolution of Enzymes." *IUCrJ* 4 (1). International Union of Crystallography:50–64.

Bauer, Paul, Alexandre Barrozo, Miha Purg, Beat Anton Amrein, Mauricio Esguerra, Philippe Barrie Wilson, Dan Thomas Major, Johan Åqvist, and Shina Caroline Lynn Kamerlin. 2018. "Q6: A Comprehensive Toolkit for Empirical Valence Bond and Related Free Energy Calculations." *SoftwareX*. Elsevier.

Fuxreiter, Monika, and Letif Mones. 2014. "The Role of Reorganization Energy in Rational Enzyme Design." *Current Opinion in Chemical Biology* 21. Elsevier:34–41.

Lee, Frederick S, Zhen-Tao Chu, Michael B Bolger, and Arieh Warshel. 1992. "Calculations of Antibody-Antigen Interactions: Microscopic and Semi-Microscopic Evaluation of the Free Energies of Binding of Phosphorylcholine Analogs to Mcpc603." *Protein Engineering, Design and Selection* 5 (3). Oxford University Press:215–28.

Maier, James A, Carmenza Martinez, Koushik Kasavajhala, Lauren Wickstrom, Kevin E Hauser, and Carlos Simmerling. 2015. "Ff14SB: Improving the Accuracy of Protein Side Chain and Backbone Parameters from ff99SB." *Journal of Chemical Theory and Computation* 11 (8). ACS Publications:3696–3713.

Marelius, John, Karin Kolmodin, Isabella Feierberg, and Johan Åqvist. 1998. "Q: A Molecular Dynamics Program for Free Energy Calculations and Empirical Valence Bond Simulations in Biomolecular Systems1." *Journal of Molecular Graphics and Modelling* 16 (4-6). Elsevier:213–25.

Muegge, Ingo, Phoebe X Qi, A Joshua Wand, Zhen T Chu, and Arieh Warshel. 1997. "The Reorganization Energy of Cytochrome c Revisited." *The Journal of Physical Chemistry B* 101 (5). ACS Publications:825–36.

Purg, Miha, and Shina Caroline Lynn Kamerlin. 2018. "Empirical Valence Bond Simulations of Organophosphate Hydrolysis: Theory and Practice." *Methods in Enzymology* 607:3–51.

Purg, Miha, Mikael Elias, and Shina Caroline Lynn Kamerlin. 2017. "Similar Active Sites and Mechanisms Do Not Lead to Cross-Promiscuity in Organophosphate Hydrolysis: Implications for Biotherapeutic Engineering." *Journal of the American Chemical Society* 139 (48). ACS Publications:17533–46.

Release, Schrödinger. 2016. "3: MacroModel." *Schrödinger, LLC, New York, NY*.

Warshel, Arieh, and Robert M Weiss. 1980. "An Empirical Valence Bond Approach for Comparing Reactions in Solutions and in Enzymes." *Journal of the American Chemical Society* 102 (20). ACS Publications:6218–26.