PCMSolver: an Application Programming Interface for the Polarizable Continuum Model

Roberto Di Remigio[1], T. Daniel Crawford[2], and Luca Frediani[1]

[1]Department of Chemistry, Centre for Theoretical and Computational Chemistry, University of Tromsø, Tromsø, Norway
[2]Department of Chemistry, Virginia Tech, Blacksburg, VA, USA

February 17, 2016

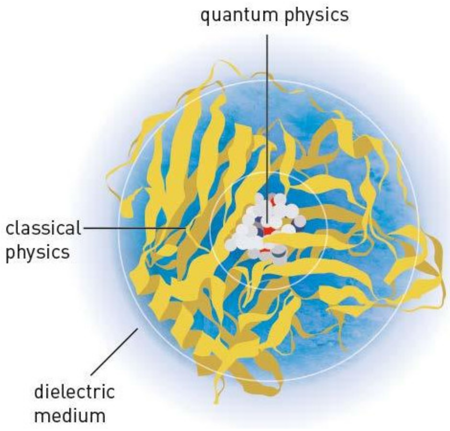
The growing complexity of quantum chemical program packages requires that an appropriate strategy be devised to implement new features. Scalability is of paramount importance, but it has become clear that maintainability and extensibility of the code play an equally important role in managing this growing complexity [1].

The use of a *modular programming paradigm* has been recognized as beneficial in many other scientific computing contexts. New features are isolated into libraries that can be accessed by host programs through a well-defined *application programming interface* (API). Computational tasks are thus implemented into separate, independent and interchangeable modules. These are *developed* and *tested* independently of the particular host program in which they will be used [2, 3]. A well-defined API clearly delimits the boundaries of the functionality offered, thus forcing a programming style and standardisation of the functionality, eventually.

When coupled with *open-source* licensing, modularity is a proper step towards ensuring reproducibility of results from scientific simulation software [4]. Open-source development can fully leverage the benefits of widespread, cloud-based, free, code development services, such as hosted distributed version control systems (DVCSs), continuous integration, code coverage, static and dynamic code analyses, nightly regression testing, public issue tracking, code review and so forth: adoption of a modern code development workflow is easily within reach.

It is, of course, true that the above mentioned services are not exclusive prerogatives of open-source projects. However, an open review process of scientific software can often help to establish reproducibility, extensibility and sustainability of the software ecosystem. Open-source software, modular development of new functionalities and full-fledged exploitation of DVCSs ensure a much larger scientific impact. Third-parties can easily contribute to the project: by improving the documentation, by reporting bugs or by actively extending the codebase with new functionality.

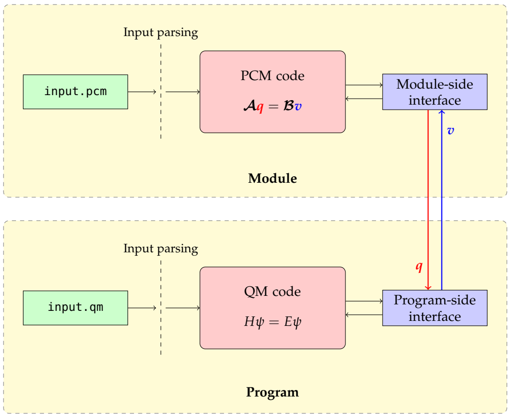
The description of chemical phenomena in complex environments, such as solutions or proteins, is as challenging as it is important for an accurate prediction of a wide range of properties [5]. Quantum chemistry is nowadays a valuable tool: its simulations can provide significant insight into complex chemical phenomena [6, 7]. The major challenge lies in providing a faithful, realistic, yet cost-effective description of the environment of the molecule: a molecule in solution can be represented by a quantum mechanical model comprising a great number of atoms, thousands or more. Such a simulation would, however, be impossible to carry out with the current state-of-the-art computational resources. Models must be devised to overcome the dimensionality "disease" and the idea of introducing different approximation levels for different parts of the system under study is particularly appealing, as it helps our chemical understanding. This is the idea at the basis of *multiscale* models. *Continuum* (or *implicit*) models, deal explicitly only with the degrees of freedom associated with the solute, while replacing the solvent with a structureless continuum characterized by its bulk properties [8]. Among these models, the polarizable continuum model (PCM) is the most widespread and successful [9, 10].



The PCM is also an ideal candidate for the creation of a solvation API. Its input from and output to *any* quantum chemistry code is limited and well-defined. This provides a natural API design: the API functions can be compared *vis-à-vis* with the working equations derived for the different quantum chemical methods.

We will elucidate the design principles adopted in the development of our `PCMSolver` library: an *open-source* API for the inclusion of the PCM in *any* quantum chemistry code [11]. Use of our API *significantly* limits coding effort on the side of the host: continuum solvation at the self-consistent field (SCF) level of theory can be implemented in the host program almost out-of-the-box. The module is agnostic of the host code internals and is tested separately from it.

We will emphasize the role of the open-source licensing model and the importance of the modern workflows adopted in our project. The adoption of `git` as distributed version control system, `CMake` for cross-platform builds and automatic documentation deployment on Read The Docs simplify extensibility of the module and promote third-party contributions to the code base, while continuous integration and nightly testing offer an invaluable level of confidence in the code. `PCMSolver` currently offers the basic PCM functionality already available in many widespread software packages, such as traditional collocation solvers for isotropic media [9]. Some unique functionalities are the wavelet Galerkin solvers on smooth molecular surfaces [12, 13], the Green's functions for spherical diffuse interfaces [14] and the delayed, time-dependent (TD) solvers for real-time TDSCF simulations [15].



Careful usage of the C++ object-oriented paradigm is the key to this wide spectrum of functionalities. The library is in itself made of modules, communicating by means of composition at the outermost level of the design. This allows us to mix languages internally: we currently have Fortran and C modules alongside the main C++ structure. Library internal classes balance the dynamic polymorphism offered by inheritance and the static polymorphism offered by template programming [16, 17]. Preeminent use of composition over inheritance keeps the coupling between different modules as low as possible [18].

As a final example of the power of our modular strategy, we will present our recent work on different quantum chemistry program packages, tackling both uncorrelated and correlated wave functions [19, 20] and also including relativistic effects [21, 22]. In particular, our recent work-in-progess on a variational reformulation of the polarizable continuum model–coupled cluster (PCM-CC) algorithm [23–25] within the `Psi4` program package will be presented with emphasis on possible applications to the prediction of the optical properties of chiral molecules in solution.

References

[1] G. Wilson et al., “Best Practices for Scientific Computing”, PLoS biology **12**, e1001745 (2014).

[2] E. W. Dijkstra, “The Structure of the ”THE”-multiprogramming System”, Commun. ACM **11**, 341 (1968).

[3] D. L. Parnas, “On the Criteria to Be Used in Decomposing Systems into Modules”, Commun. ACM **15**, 1053 (1972).

[4] J. D. Gezelter, “Open Source and Open Data Should Be Standard Practices”, J. Phys. Chem. Lett. **6**, 1168–1169 (2015).

[5] C. Reichardt and T. Welton, *Solvents and Solvent Effects in Organic Chemistry* (Wiley-VCH Verlag GmbH & Co. KGaA, 2010).

[6] Nobel Media AB 2014, *The Nobel Prize in Chemistry 1998*, (1998) `http://www.nobelprize.org/nobel_prizes/chemistry/laureates/1998/`.

[7] Nobel Media AB 2014, *The Nobel Prize in Chemistry 2013*, (2013) `http://www.nobelprize.org/nobel_prizes/chemistry/laureates/2013/`.

[8] L. Onsager, “Electric Moments of Molecules in Liquids”, J. Am. Chem. Soc. **58**, 1486 (1936).

[9] J. Tomasi, B. Mennucci, and R. Cammi, “Quantum Mechanical Continuum Solvation Models”, Chem. Rev. **105**, 2999 (2005).

[10] B. Mennucci and R. Cammi, eds., *Continuum Solvation Models in Chemical Physics* (John Wiley & Sons, Ltd, 2007).

[11] `PCMSolver`, an Application Programming Interface for the Polarizable Continuum Model, written by R. Di Remigio, L. Frediani and K. Mozgawa, (see `http://pcmsolver.readthedocs.org`).

[12] V. Weijo, M. Randrianarivony, H. Harbrecht, and L. Frediani, “Wavelet Formulation of the Polarizable Continuum Model”, J. Comput. Chem. **31**, 1469–1477 (2010).

[13] M. Bugeanu, R. Di Remigio, K. Mozgawa, S. S. Reine, H. Harbrecht, and L. Frediani, “Wavelet Formulation of the Polarizable Continuum Model. II. Use of Piecewise Bilinear Boundary Elements”, Phys. Chem. Chem. Phys. (2015) `10.1039/C5CP03410H`.

[14] K. Mozgawa, R. Di Remigio, and L. Frediani, “A Polarizable Continuum Model for Molecules at Spherical Diffuse Interfaces”, In preparation.

[15] S. Corni, S. Pipolo, and R. Cammi, “Equation of Motion for the Solvent Polarization Apparent Charges in the Polarizable Continuum Model: Application to Real-Time TDDFT”, J. Phys. Chem. A **119**, 5405–5416 (2015).

[16] A. Alexandrescu, *Modern C++ Design: Generic Programming and Design Patterns Applied* (Addison-Wesley Longman Publishing Co., Inc., 2001).

[17] H. Sutter and A. Alexandrescu, *C++ Coding Standards: 101 Rules, Guidelines, and Best Practices (C++ in Depth Series)* (Addison-Wesley Professional, 2004).

[18] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software* (Addison-Wesley Longman Publishing Co., Inc., 1994).

[19] K. Aidas et al., “The Dalton Quantum Chemistry Program System”, Wiley Interdiscip. Rev. Comput. Mol. Sci. **4**, 269–284 (2014).

[20] J. M. Turney et al., “Psi4: An Open-Source Ab Initio Electronic Structure Program”, Wiley Interdiscip. Rev. Comput. Mol. Sci. **2**, 556–565 (2012).

[21] DIRAC, a relativistic ab initio electronic structure program, Release DIRAC14 (2014), (see `http://www.diracprogram.org`).

[22] R. D. Remigio, R. Bast, L. Frediani, and T. Saue, “Four-Component Relativistic Calculations in Solution with the Polarizable Continuum Model of Solvation: Theory, Implementation, and Application to the Group 16 Dihydrides H_2X (X = O, S, Se, Te, Po)”, J. Phys. Chem. A **119**, 5061–5077 (2015).

[23] R. Cammi, “Quantum Cluster Theory for the Polarizable Continuum Model. I. The CCSD Level with Analytical First and Second Derivatives”, J. Chem. Phys. **131**, 164104 (2009).

[24] F. Lipparini, G. Scalmani, B. Mennucci, and M. J. Frisch, “Self-Consistent Field and Polarizable Continuum Model: A New Strategy of Solution for the Coupled Equations”, J. Chem. Theory Comput. **7**, 610 (2011).

[25] M. Caricato, G. Scalmani, G. W. Trucks, and M. J. Frisch, “Coupled Cluster Calculations in Solution with the Polarizable Continuum Model of Solvation”, J. Phys. Chem. Lett. **1**, 2369 (2010).