A New Algorithm for Second Order Perturbation Theory

Andrey Asadchev Mark S. Gordon

1 Conclusions

As the computing technology changes and matures, the scientific computing must follow. The hardware and software which was cutting edge in the 70's and 80's still dictates how many of the computational chemistry packages are still implemented today. However, the computing technology evolved very quickly since the Fortran 77. Object oriented programming (OOP), generic programming, standard libraries, and system standards have become the essential pieces of most modern commercial and open-source software, small and large alike. To keep up with the improvements in computer science, the computational chemistry algorithms must be either modernized or rewritten. Often, due to software architecture decisions made decades ago, rewriting is the only viable plan for the future. Not all of the software needs to be modernized at once: the key pieces such as integral and Hartree-Fock methods can be rewritten alone and integrated into the existing software, one at a time. Software modernization also presents a opportunity to improve the existing algorithms, separate them into modular libraries to encourage reuse among the scientists, and to plan ahead, given the trends in computing over the last few decades.

The first algorithm presented was the Hartree-Fock, the reference method in the most election correlation theories. The Hartree-Fock method requires evaluation of the two-electron integrals, which constitutes the most consuming part. Unlike other pieces in the computational chemistry, two-electron integral methods are specific to the domain and do not receive much attention from outside the field. The integrals were implemented using Rys Quadrature, one out of several integral methods. While algorithmically more complex than other methods, Rys Quadrature is a general numerically stable method with low memory footprint, which makes it suitable for implementation on graphical processing units (GPU). Once the integral engine was implemented, the multithreaded Hartree-Fock method naturally followed. The integral and Hartree-Fock GPU implementation was able to reuse many key pieces of the CPU algorithm, designed to be fast, extensible, and flexible through the use of code generator and C++ templates.

One of the most common electron correlation methods is second order many-body perturbation theory, also known as Moller-Plesset second order perturbation theory (MP2). Unlike higher-order treatments, MP2 is relatively inexpe black-box method which makes it very popular. Hence, the Hartree-Fock implementation was followed by MP2 method. Like the Hartree-Fock method, the MP2 implementation relies heavily on the fast integrals. But unlike the Hartree-Fock, most of computational work is handled by the de facto standard basic linear algebra subroutines, BLAS. The MP2 algorithm implemented is a semi-direct method, meaning that the partially transformed integrals need to be stored in the secondary storage, such as disk or distributed memory. Unlike the other MP2 algorithms, which are either disk or distributed memory, the implemented algorithm uses OOP features of C++ to provide transparent integral storage on either disk or in distributed memory.

The natural follow-up to MP2 is the coupled cluster (CC) theory. The couple cluster truncated at singles and doubles excitations, CCSD, with the perturbative triples correction (T) leads to CCSD(T) method, often called the gold standard of computational chemistry due its accuracy. The CCSD(T) is very expensive method, both in terms of computer time and memory. However, with the lessons learned designing the MP2 algorithm, a fast CCSD(T) algorithm was developed such that it could run equally a single workstations and supercomputers. The key to implementation was optimizing the algorithm in terms of memory first, I/O overhead second, and concentrating on the computational efficiency last. By using several properties of atomic to molecular basis transformations, several expensive computation and storage requirements were eliminated from the CCSD algorithm. And by using well-known loop optimization technique called blocking, the (T) algorithm was implemented with very little memory and very little I/O overhead.

The three algorithms summarized above were prompted by the need to accommodate the wide array of computational hardware. In the process, the algorithms were improved, often drastically. Implemented in C++, the algorithms and the supporting framework were built as a stand-alone library, with Fortran bindings. Connected to GAMESS, the library was successively integrated with the existing legacy code. While not explicitly discussed, the supporting framework, such as basis set and wavefunction objects, is absolutely necessary to develop robust flexible modern code.