Making Innovation Easier in Quantum Many-Body Problem

Edward F. Valeev
Department of Chemistry, Virginia Tech, Blacksburg, VA 24061

What Quantum Chemists/Physicists Need From Programming Environments

Many-body quantum mechanics (QM) is at the heart of modern chemistry and materials science, by describing the fundamental particle behavior that governs phenomena from smallest (e.g. chemical bond) to the largest (e.g. superconductivity) scales. The computational challenge of the many-body QM is threefold: potentially exponential — in practice, high-order polynomial — complexity in computation and storage, challenging numerics (large discretization error, nonlinearities, etc.), and high complexity of equations. Many methods exist for solving the quantum many-body problem: density-functional theory, coupled-cluster, quantum Monte Carlo, renormalization group, etc. Despite their successes in present form it is clear that these methods have serious limitations and thus are being improved constantly.

It is a reasonable guess that high long-term impact on the area of many-body simulation will come from new ideas for the many-body formalism, numerical representation, and algorithms. See for evidence the huge impact of the formal and algorithmic advances on the quantum chemistry in the past two decades, even though the basic way we develop software has changed relatively little. Thus, the programming environment community can make the highest impact on our field

- by focusing not on particular existing methods, but on the computation concepts and patterns common to the existing methods, and
- by enabling rapid development of new methods, whether in or outside the context of the vast existing legacy software landscape, at full computation scale afforded by the modern computers.

Tensor Algebra as The Language of Quantum Many-Body Physics

The basic ingredients of many-body QM are tensors (or just multidimensional arrays) representing functions and operators. The integro-differential calculus in many dimensions is replaced with tensor algebra. Due to the exponential complexity of the exact solutions, these tensors are usually sparse, whether in a model-specific sense (e.g. the wave operator in the coupled-cluster is restricted to some subset of "excitation ranks") or more general mathematical sense (rank and block/element sparsity). Although multidimensional arrays are supported by all general-purpose languages currently used for numerical computing, the level of composition — explicit index loops — is rather low outside of a few trivial operations. Support for *high-level* expression of tensor algebra, preferably for sparse distributed data, is needed. We should note specifically Global Arrays Toolkit as a *library* implementation of a limited subset of distributed tensor algebra, and Chapel and X10 as examples of HPC languages that support distributed Arrays and composition of user-defined algorithms on them. However, all of these tools are limited and the development of tensor algebra is usually a custom job that's rarely, if ever, gets leveraged outside the immediate development circle. Example: pretty much any quantum chemistry

package (for a few exceptions see libtensor library of Epifanovsky and Cyclops Tensor Framework of Solomonik).

Lack of common tensor algebra tools is not all bad; our understanding of how to represent and compute many-body QM equations is incomplete, evolves in time, and cannot be supported by a single field-wide implementation. What is bad is how complicated it is to develop these tools. Another challenge is the lack of established best-practices concepts for how to represent and operate on distributed, and especially sparse, tensors. We (quantum many-body theorists) need help with these, from the programming environments community and, more broadly, from the computer science and mathematics communities.

Relevant Topics for Discussion

To start a conversation among the diverse group of attendees of this workshop, here are a few questions, roughly divided by topics.

- How do we envision the process of converting ideas (equations) into software in 10 years? What level of composition do we need for novices (graduate students) and experts (PIs)? Do we need languages or language extensions; can we instead get away with libraries? How do we enable customization of syntactic constructs to fit the particular domain?

- How would we like to compose tensor algebra? What types of sparsity (block, element, rank) and hierarchical organization (chains as in DMRG, trees in MRA) do we want to represent? Is there a way to abstract tensor operations from the data, a la standard algorithms in C++? What distributed data structures can represent these diverse types of tensors? What distributed operations are needed to represent algorithms for distributed tensors? What are the relevant building blocks for these algorithms, Message Passing, etc.? How can construction of user-defined algorithms be supported?

Many of these questions are research topics in themselves, but they can be used to identify the areas of biggest need and highest potential impact from the collaboration between the domain scientists and the programming environments community.

Afterword

These opinions are informed by 20 years of software development in the area of many-body quantum simulation in the context of molecular quantum chemistry. The author is a major contributor to several open- (Libint, MPQC, MADNESS, TiledArray, Psi) and closed-source (ORCA) software packages. Much of the current thinking on this topic is informed by the development of TiledArray, a C++ library for concurrent block-sparse tensor computation.