Software tools for the future: perspective from quantum Monte Carlo (QMC) – white paper March 28-30, 2014

L. Mitas, North Carolina State University, Raleigh

Introduction.

Quantum Monte Carlo methods are based on three key ingredients:

i) approximate analytical and explicit constructions of solutions for fundamental, many-body, quantum equations: stationary Schrodinger eq. for eigenstates and eigenenergies, Bloch equation for temperature density matrix, etc. This raises the importance of sophisticated and explicitly many-body constructions: they are often decisive for both the physical insights and the computational efficiencies.
ii) stochastic algorithms of solving the fundamental equations, ie, uses random walks, etc.
iii) computer performance (mostly on the capacity front).

Each of these three aspects is almost equally important in solving many-body problems of interest. Additional goal for QMC is the ability of solving very diverse set of problems that often cover broad areas of condensed matter physics. Some examples, beyond more common molecular and solid state electronic structure systems:

a) nanosystems with artificial geometries (variety of quantum dots etc)
b) systems with external fields and "exotic" quantum phenomena (quantum Hall effect etc)
c) systems with effective interactions such as unitary fermions in ultracold condensates
d) paradigmatic model systems such as electron gas, etc.

Therefore, the versatility and adaptability of the codes to new areas is much more important and in higher demand than perhaps it is for most of the quantum chemistry codes. Parallelization: although QMC appears to be "embarrassingly parallel", as soon as one hits some of the limits with memory/cash sizes due to size of the system, for example, parallelization of the QMC code becomes non-trivial. Still, it seems that so far it is less complicated than for many other problems that do not have an "easy parallelization axis". QMC has a number of these (say, random walkers, basis functions, electrons, etc) so that MPI, OpenMP, etc, is enough for useful classes of applications and one can reach high efficiencies on tens of thousands of cores. Note also that QMC is not married to one type of basis set as are many quantum chemical methods. Just the opposite: one can use almost any type of basis functions, their combinations, etc.

Hardware

As it is clear from new hardware and architectures the new high-performance platforms are build on hierarchies of nodes/cores/accelerators with corresponding and varying hierarchies of memory. Probably there is not much we can do about this except to adapt.

Software: simple pieces, nimble constructions

Research software of future should be more adaptable, nimble and capable of rapid adjustment to new research problems and opportunities. This would also resolve the questions whether it can be maintained in future, whether the project will be alive in a few years, etc: only the components that are useful and that are used further would survive. We can and should think about the scientific software tools as a Lego boxes constructions where the size of each component has (only) a manageable amount of complexity and functionality. The interfaces between the components should be more clearly defined, perhaps even enforced as “contracts”, akin to constructions, say, in Eiffel. They should be also able to access variety of data sources (beyond files) including web, say, during the run time. *An interesting question is what would be what the right software framework for such type of project.* I hope to learn something new in this regard at the workshop.

What we also need are libraries/software to deal with hierarchies of memories and compute nodes/processors/accelerators in an easy, transparent and unified way, say, in C/C++. That would be extremely useful, perhaps some of that already exists, but I doubt that it is broadly applicable. No matter how fast the accelerators are “down there”, in research one always has to make the following decision: how much of a new science I can produce by recoding/exploiting the latest accelerators vs. by trying a new physical/analytical/algorithmic idea. Note that the second option often wins. Just to mention the famous quip by Jim Chelikowsky: if you can choose between yesterday’s algorithms on today’s hardware **vs** today’s algorithms on yesterday’s hardware, which one would you choose ? Almost always the latter one. *This could possibly change providing useful and simple to use libraries for new hardware would be available. (It seems that computer science is always in a reactive mode in that regard. Can’t they* ***be ahead*** *of the hardware ?)*

I would be also very much interested in translators from Python to C/C++. I know about Cython, ShedSkin, py2cpp, etc. All these appear to be sort of “half-way” or not there yet. Something that is reasonably close to “almost fully done”. (Perhaps too special for my interests).

Needless to say, everything should be in Open Source, it is tremendously convenient (at least for us).

Role of data

One of the things that is clearly missing are well-organized data produced by the codes we have in mind. Well-designed methods should generate data that is easy to understand, **it is compact** and should be generated automatically with metadata that is web/github/other-storage storable, accessible and readable by other codes. We need an ecology of the data that can be consumed by the “code animals”. I do not have necessary in mind so-called “big data”, in fact something almost opposite. Big data is produced by processes or sampling where the underlying system is so complex that we do not know the fundamental equations or they are impractical to apply. Therefore in big data we want to understand or dig out something useful from giant piles of some unremarkable data points and noise. Our case is very different: we know the fundamental equations, we are gradually getting the fundamentals of many-body effects and therefore we should produce *compact data/description of the systems, phenomena and properties* we study. Very much in the spirit of Schrodinger equation being the most compact formulation of the needed piece of quantum mechanics. What should be there is the *minimal amount* of data that enables to launch new calculations building upon previously achieved results or to reproduce it with appropriate/reasonable amount of effort. That means: well-organized, tested, benchmarked with metadata that enable to access from variety of codes, on-the-fly.