Time-dependent density functional theory applied to superfluid nuclei

Aurel Bulgac\textsuperscript{1} and Kenneth J Roche\textsuperscript{2}

\textsuperscript{1}Department of Physics, University of Washington, Seattle, WA 98195-1560, USA
\textsuperscript{2}Computer Science and Mathematics Division, Oak Ridge National Laboratory, Oak Ridge, TN 37831-6173, USA

E-mail: bulgac@phys.washington.edu and rochekj@ornl.gov

Abstract. We describe the response of superfluid nuclei to any external time-dependent probe within an extension of the density functional theory, the time-dependent superfluid local density approximation (TD-SLDA). All quasi-particle wave functions (qpwf\textsubscript{s}) are represented on a three-dimensional spatial grid to allow for the breaking of all possible symmetries, and subsequently these functions are evolved in time using a high order multi-step algorithm. Mathematically this problem is a system of coupled, time-dependent, nonlinear partial differential equations, which when discretized becomes equivalent to a nonlinear classical mechanics problem with \( \propto N_x^6 \approx 10^{12} \) degrees of freedom, where \( N_x \) is number of spatial mesh points in one direction. The lattice representation of the qpwfs allows for fast and accurate evaluation of various spatial derivatives required for the construction of various physical quantities that appear in the theory. The total number of independent qpwfs is of the order of spatial mesh points, and the system is evolved for tens to hundreds of thousands of time steps. We describe how we implemented the TD-SLDA method and report some of the challenges encountered.

1. Introduction

In condensed matter physics and quantum chemistry density functional theory (DFT) and time-dependent density functional theory (TD-DFT) are commonly used approaches to describe the ground state properties of electronic systems [1], otherwise known as normal fermionic systems. However, there is an extended class of physical systems for which these are inappropriate tools, namely, the superfluid systems [2], the most common electronic incarnation of which are the superconductors. While for normal systems the number of single-particle wave functions needed in a (TD-)DFT description is equal to the number of particles, in the case of superfluid systems this number could exceed the number of particles by a very large factor. The larger the critical temperature of the system, the larger this factor is and the more demanding the computational complexity becomes. Superfluids, for which the pairing gap is sufficiently large when compared to the Fermi energy of the system, can be encountered in a wide range of situations, such as fermionic cold atoms in traps, high \( T_c \)-superconductors, nucleons in nuclei and neutron stars, and quarks in high-density QCD matter.
2. TD-SLDA

Mathematically the TD-SLDA requires the solution of a large set of coupled time-dependent nonlinear partial differential equations:

\[
i \frac{\partial \psi_n(\vec{r}, t)}{\partial t} = \left[ h(\vec{r}, t) + V_{\text{ext}}(\vec{r}, t) \right] \psi_n(\vec{r}, t),
\]

where \(i\) is the imaginary unit, \(\psi_n(\vec{r}, t)\) are complex (wave) functions of time and 3D space coordinates, and each of them is labeled by an index \(n\) that takes usually both discrete and continuous values. \(h(\vec{r}, t)\) is a 3D partial differential operator depending in a nonlinear manner on the entire set of complex functions \(\psi_n(\vec{r}, t)\) (a feature routinely referred to as self-consistency), and \(V_{\text{ext}}(\vec{r}, t)\) is an arbitrary function of time and 3D space coordinate that models a specific external probe of the system. Spin and isospin coordinates are not displayed for simplicity of presentation. Typically \(h(\vec{r}, t) + V_{\text{ext}}(\vec{r}, t)\) is a Hermitian operator, and for that reason the above system of nonlinear partial differential equations has an infinite number of conserved quantities, but that, however, does not make it an integrable system. After time-dependent solutions have been computed, one can extract various densities and through a time-dependent Fourier transform extract physical observables of interest. The number of various densities needed is of order of ten or so in most cases.

3. Implementation of TD-SLDA

For the studies we envision, in principle 3,000 to 10,000 nuclei are ultimately of interest. Simple estimates show that to extract the observables with a reasonable accuracy and over a frequency interval of interest for a particular nucleus, of the order \(10^4\) to \(10^5\) time steps must be performed for a given nucleus.

Taking a time step means solving equation (1). In order to solve these equations the system is placed on a 3D spatial lattice with a total \(N_{3D} = N_x \times N_y \times N_z\) mesh points. The size of the lattice and the prescribed lattice spacing are calibrated according to the total number of nucleons of the nucleus to be studied. The number of complex functions \(\psi_n(\vec{r}, t)\) at a given time is \(O(N_{3D})\), that is, the number of labels \(n\) of the qpwfs. Each of these functions has an internal complexity that requires additional indices to label spin, isospin, and particle-hole degrees of freedom.

Here we explicitly compute the variation of the energy of the system on the lattice. The first action our code takes after some system diagnostics is that the lattices in both coordinate and momentum space are initialized. Next, the kinetic energy is evaluated. This process is simplified by the fact that often this term is diagonal in k-space in our chosen representation. Also, the potentials have to be initialized at each point on the coordinate lattice. In the nuclear problem this includes central, Coulomb, spin-orbit, and pairing potentials. From the qpwfs we construct various densities, which are bilinear combinations of the qpwfs, (e.g., the proton and neutron number densities, anomalous, kinetic energy, spin-orbit densities) on the lattice at each coordinate. With the densities initialized and the potential initialized, the system energy and the particle numbers of the system are evaluated.

The densities are bilinear combinations of the qpwfs, the complex conjugates of the qpwfs, and of the various spatial partial derivatives of the qpwfs. The plane wave lattice representation allows for fast and accurate computation of partial derivatives. The operation is simply two discrete Fourier transforms and an element-by-element array multiply per qpwf. For 3D lattices of up to \(128^3\) it was shown on various architectures that an FFTW [3] evaluation of partial derivatives is about a factor of seven at most the time required for a simple multiplication of two 3D arrays element-by-element [4]. If the lattice spacing is chosen appropriately for each physical system to be studied, the accuracy of this representation of the wave functions can easily attain almost machine precision [5].
To perform the time evolution we have chosen a multistep predictor-modifier-corrector method due to Adams-Bashforth-Milne (ABM) [6]. The predictor and modifier are calculated for each qpwfs at each lattice site. This computation is a function of the qpwfs and their time derivatives at multiple time steps and includes operations such as discrete Fourier transforms. The modified densities are computed, and these terms are subsequently used to modify the potentials. From the modified densities and potentials, the corrector for each qpwf is calculated at each site. The corrected densities and potentials are constructed, total energy and total particle number are evaluated, and time is officially iterated a time step. Other methods can be easily implemented, but this choice has some advantages, not necessarily unique to this method. The ABM algorithm requires only two rhs evaluations per time step, it is a fifth-order method and is apparently also very stable numerically for long time intervals for the class of problems we envision studying. Take a $40^3$ lattice as an example, the system energy and number density are numerically conserved, showing a variance of $O(1/10^6)$ over a thousand time steps during the evolution of 66796 qpwfs. The algorithm has also been tested on a simpler problem, a time-dependent homogeneous unitary Fermi gas, where a system of about 30,000 differential equations was integrated for about 300,000 time steps [7].

A major drawback of multi-step algorithms is the heavy memory requirements. This problem can be managed a number of ways but we are pursuing two particular approaches: out-of-core and aggregated distributed memory.

In the out-of-core technique, it is assumed that not enough resources are available to store in memory all the data required to compute system observables within a time step. Thus, the information required to describe each quasi-particle state is either stored in memory or on disk in files. To evolve the system requires read and write operations on files stored on disk. This approach suffers from the poor performance of swapping data via disk accesses during each phase of computation. It has the advantage that larger lattices can be used to study the system. For instance, $N_x = N_y = N_z = 100$ lattice yields 1046604 qpwfs and would require $\sim 800 \times 2^{40}$ bytes of memory to run the problem without access to the filesystem. With swapping, this system can be computed on a single node allocation where there are $4 \times 2^{30}$ bytes of RAM. For lattices greater than $130^3$, one is forced to reconsider the out-of-core approach since even swapping with the current data structures is unwieldy to allocating memory for a base unit of work. However, for realistic problems ($40^3$ or greater) the operational complexity is $O(2^{40} \text{ flop})$ or greater per time step, and so the return time of the calculation would be unacceptable not just because of time spent swapping, but also because of time spent computing the work that goes into computing a physical time step.

The aggregated distributed-memory approach essentially requests the number of compute nodes sufficient to fit the entire problem in physical memory and then relies on network communication to organize the computation. There is some redundancy not present in the sequential code implementation or an out-of-core scheme since each process in the distributed memory approach needs a copy of the lattices as well as global system observables. The mapping that we use to reduce both data and instruction complexity relates MPI [8] process ranks to disjoint ranges of canonical labels of the qpwfs. Each process manages the same number of functions and all other data derived from those functions within a time step (differing by at most a single qpwf between MPI processes). Thus, both the storage and instruction complexity are balanced per process. In order to compute the various densities of the system, each process simultaneously evaluates the partial contribution of the qpwfs it manages locally on its copy of the lattices. The complete densities are then constructed by communication over the distributed network via a global reduction of these partial values at each point of the lattice followed by a broadcast of the total densities. The updated densities of the system are used to modify the system potentials and construct the values of the qpwfs at the next time step. This approach is still memory bound as the lattice dimensions grow. For instance, as of today and without any
modifications, the current code cannot study lattices larger than \( \sim 60^3 \) on the 7832 Quad Core AMD Opteron [9] compute nodes of the DOE’s Cray XT [10] system in the National Center for Computational Sciences at Oak Ridge National Laboratory [11]. This equates to evolving 226156 qpwfs, a number that may already be more than sufficient to study most nuclei of interest.

The discretization described here also has the advantage that it exhibits outstanding strong scaling trends; see table 1 for an example. The idea of weak scaling for an algorithm is that we increase the problem complexity by some factor, increase the PE count by the same factor, and the runtime remains constant. We are studying \[ \text{cost (\$ \text{, time, instructions, etc.) / lattice site / qpwf / PE / time step}} \] as a weak scaling metric. Tables 2 and 3 provide reference material for the weak scaling behavior of the code on jaguar.ccs.ornl.gov.

Table 1. Strong scaling on dual-core AMD Opteron Cray XT. The problem complexity is fixed at 28288 qpwfs on a \( 30^3 \) lattice for a single time step. MPI processes (PEs) are added thus reducing the storage and work load per process. The time reported is the wall-time in seconds expired to compute the single time step for a given number of processes.

| PEs    | 576 | 1152 | 1728 | 2304 |
|--------|-----|------|------|------|
| NWF/PE | 48  | 24   | 16   | 12   |
| \[s\]/TS| 56.2| 28.8 | 19.3 | 14.92|

Table 2. Hardware event profile data from Quad Core AMD Opteron(tm) Cray XT for 10 time steps of TD-SLDA evolution.

| \( N_x^3 \) | \( 30^3 \) | \( 40^3 \) | \( 50^3 \) |
|------------|------------|------------|------------|
| QPWF       | 28288      | 66796      | 130528     |
| PEs        | 168        | 942        | 3626       |
| \[s\]      | 297.31     | 296.15     | 319.03     |
| INS        | 141538488856847 | 787635078990076 | 3133849257252106 |
| FLOP       | 33786997017008 | 184227473630036 | 822772291175810 |

The current software implementation of the TD-SLDA discretization exists in both the Fortran and C programming languages. Distributed memory parallel computing systems designed with both Intel Xeon(tm) and AMD Opteron(tm) multicore chipsets have been successfully tested to conserve significant system observables as a function of time. The GNU Compiler Collection (GCC) [12], Portland Group (PGI) [13], and Intel [14] compilers have been used during testing on these systems. The FFTW and MPI [8] libraries have been used and the code employs the PAPI api [15] to provide chip-level diagnostics of the hardware during code execution. We plan to pursue a hybrid version of the code capable of exploiting the benefits of both the out-of-core and aggregated distributed memory techniques.

Acknowledgments
We thank Y. Yu for major help at the initial stages of coding the theory and the UNEDF SciDAC collaboration for support through the US Department of Energy under grant No. DE-FC02-07ER41457. This research is sponsored by the Office of Advanced Scientific Computing Research; U.S. Department of Energy. The work was performed at the Oak Ridge National Laboratory, which is managed by UT-Battelle, LLC under Contract No. De-AC05-00OR22725.
Table 3. Weak scaling assertion of TD-SLDA on Quad Core AMD Opteron(tm) Cray XT (reference Table [2]). The ratios of the runtimes are reported in column 1. These should be 1 in the ideal case. Column two is the weak scaling factor as it is the ratio of the more complex problem hardware allocation to the less complex problem hardware allocation. Column 3 is the ratio of the total executed instructions for the more complex problem to the total executed instructions of the less complex problem. Ideally, this number would be greater than or equal to the number in column 2. Since this is not ideal, remember to skew the result by the value in column 1. Column 4 is the ratio of the total floating point instructions executed for the more complex problem to the total floating point instructions executed by the less complex problem. Again, we expect this number to be consistent with the weak scaling factor. Thus, for these runs, the weak scaling of the TD-SLDA code is not perfect but is basically excellent.

|       | PE[50]\textsuperscript{3} | PE[40]\textsuperscript{3} | INS[50]\textsuperscript{3} | INS[40]\textsuperscript{3} | FLOP[50]\textsuperscript{3} | FLOP[40]\textsuperscript{3} |
|-------|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|
| \(T[50]\textsuperscript{3}\) | 1.07(7)                  | 3.84(9)                  | 3.97                     |                         | 4.46                     |
| \(T[40]\textsuperscript{3}\) | 1.07                     | 21.58                    | 22.14                    |                         | 24.35                    |
| \(T[30]\textsuperscript{3}\) | 0.996                    | 5.60(7)                  | 5.56                     |                         | 5.45                     |

References
[1] Dreizler R M and Gross E K U 1990 Density Functional Theory: An Approach to Quantum Many-Body Problem (Berlin: Springer).
[2] Bulgac A 2007, Phys. Rev. A 76 040502(R).
[3] The Fastest Fourier Transform in the West, www.fftw.org
[4] Aichinger A and Krotscheck E 2005, Comp. Mat. Science 34 188.
[5] Littlejohn R G, Cargo M, Tucker Carrington J, Mitchell K A, and Poirier B, 2002, J. Chem. Phys. 116 8691.
[6] Hamming R W 1962 Numerical Methods for Scientists and Engineers, (New York: McGraw-Hill).
[7] Bulgac A and Yoon S 2008, in preparation.
[8] Message Passing Interface, www.mpi-forum.org
[9] Advanced Micro Devices (AMD), Software Optimization Guide for AMD Family 10h Processors, rev. 3.0.6 (2008); Compiler Usage Guidelines for AMD64 Platforms, rev. 3.22 (2007).
[10] Cray Inc., www.cray.com
[11] National Center for Computational Sciences, www.nccs.gov
[12] The GNU Project, www.gnu.org
[13] The Portland Group Compilers, www.pgroup.com
[14] Intel, developer.intel.com
[15] Performance Application Programming Interface, icl.cs.utk.edu/papi