Project 2, Variational Monte Carlo studies of helium and beryllium

The final aim of this project is to develop a variational Monte Carlo program which can be used to obtain ground state properties of atoms like He, Be, O, Ne, Si etc as well as diatomic molecules. for important molecules

The aim of the second project is to parallelize the code from project 1 and implement an efficient algorithm for computing the Slater determinant. We will apply the new algorithm for the Slater determinant as well as a proper parallelization to the ground state of beryllium and neon.

The deadline for project 2 is March 27, at noon. See below for delivery format.

Variational Monte Carlo calculations of the beryllium and the neon atoms

Project 1 has prepared you for extending your calculational machinery to other systems. Here we will focus on the beryllium and neon atoms. It is convenient to make modules or classes of trial wave functions, both many-body wave functions and single-particle wave functions and the quantum numbers involved, such as spin, orbital momentum and principal quantum numbers.

If we stick to hydrogen-like wave functions, the trial wave function for Beryllium can be written as

$$\psi_T(\mathbf{r_1},\mathbf{r_2},\mathbf{r_3},\mathbf{r_4}) = Det\left(\phi_1(\mathbf{r_1}),\phi_2(\mathbf{r_2}),\phi_3(\mathbf{r_3}),\phi_4(\mathbf{r_4})\right)\prod_{i<j}^{4}\exp\left(\frac{r_{ij}}{2(1+\beta r_{ij})}\right), \tag{1}$$

where Det is a Slater determinant and the single-particle wave functions are the hydrogen wave functions for the $1s$ and $2s$ orbitals. Their form within the variational ansatz are given by

$$\phi_{1s}(\mathbf{r_i}) = e^{-\alpha r_i}, \tag{2}$$

and

$$\phi_{2s}(\mathbf{r_i}) = (1-\alpha r_i/2)\, e^{-\alpha r_i/2}. \tag{3}$$

For the correlation part

$$\Psi_C = \prod_{i<j} g(r_{ij}) = \exp\left\{\sum_{i<j}\frac{a r_{ij}}{1+\beta r_{ij}}\right\},$$

we need to take into account whether electrons have equal or opposite spins since we have to obey the electron-electron cusp condition as well. For Beryllium, as an example, you can fix electrons 1 and 2 to have spin up while electrons 3 and 4 have spin down. When the electrons have equal spins

$$a = 1/4,$$

while for opposite spins (as for the ground state of helium)

$$a = 1/2.$$

For neon, the trial wave function can take the form

$$\psi_T(\mathbf{r_1},\mathbf{r_2},\ldots,\mathbf{r_{10}}) = Det\left(\phi_1(\mathbf{r_1}),\phi_2(\mathbf{r_2}),\ldots,\phi_{10}(\mathbf{r_{10}})\right)\prod_{i<j}^{10}\exp\left(\frac{r_{ij}}{2(1+\beta r_{ij})}\right), \tag{4}$$

In this case you need to include the $2p$ wave function as well. It is given as

$$\phi_{2p}(\mathbf{r_i}) = \alpha\mathbf{r_i}e^{-\alpha r_i/2}. \tag{5}$$

Observe that $r_i = \sqrt{r_{i_x}^2 + r_{i_y}^2 + r_{i_z}^2}$.

(a) Write a function which sets up the Slater determinant for beryllium and neon. Compute the ground state energies for beryllium and neon as you did in project 1. The calculations should include blocking and importance sampling using the closed form expression for the local energy.

(b) The next step is to parallelize your code using either OpenMP or MPI. Test whether your code as an optimal speed-up or not. Implement unit tests as well in your program.

c) With the optimal parameters for the ground state wave function, compute again the onebody density for beryllium. Compute the onebody density for neon as well. Discuss your results and compare the results with those obtained with a pure hydrogenic wave functions. Run a Monte Carlo calculations without the Jastrow factor as well and compute the same quantities. How important are the correlations induced by the Jastrow factor?

How to write the report

What should the report contain and how can I structure it? A typical structure follows here.

- An abstract with the main findings.
- An introduction where you explain the aims and rationale for the physics case and what you have done. At the end of the introduction you should give a brief summary of the structure of the report
- Theoretical models and technicalities. This sections ends often being the methods section.
- Results and discussion
- Conclusions and perspectives
- Appendix with extra material
- Bibliography

Keep always a good log of what you do.

What should I focus on? Introduction.

You don't need to answer all questions in a chronological order. When you write the introduction you could focus on the following aspects

- Motivate the reader, the first part of the introduction gives always a motivation and tries to give the overarching ideas
- What I have done
- The structure of the report, how it is organized etc

What should I focus on? Methods sections.

- Describe the methods and algorithms
- You need to explain how you implemented the methods and also say something about the structure of your algorithm and present some parts of your code
- You should plug in some calculations to demonstrate your code, such as selected runs used to validate and verify your results. The latter is extremely important!! A reader needs to understand that your code reproduces selected benchmarks and reproduces previous results, either numerical and/or well-known closed form expressions.

What should I focus on? Results sections.

- Present your results
- Give a critical discussion of your work and place it in the correct context.
- Relate your work to other calculations/studies
- An eventual reader should be able to reproduce your calculations if she/he wants to do so. All input variables should be properly explained.
- Make sure that figures and tables contain enough information in their captions, axis labels etc so that an eventual reader can gain a first impression of your work by studying figures and tables only.

What should I focus on? Conclusions sections.

- State your main findings and interpretations
- Try as far as possible to present perspectives for future work
- Try to discuss the pros and cons of the methods and possible improvements

What should I focus on? Additional material, appendices.

- Additional calculations used to validate the codes
- Selected calculations, these can be listed with few comments
- Listing of the code if you feel this is necessary
- You can consider moving parts of the material from the methods section to the appendix. You can also place additional material on your webpage.

What should I focus on? References.

- Give always references to material you base your work on, either scientific articles/reports or books.
- Refer to articles as: name(s) of author(s), journal, volume (boldfaced), page and year in parenthesis.
- Refer to books as: name(s) of author(s), title of book, publisher, place and year, eventual page numbers

Format for electronic delivery of report and programs

Your are free to choose your format for handing in. The simplest way is that you send us your github link that contains the report in your chosen format(pdf, ps, docx, ipython notebook etc) and the programs. As programming language you have to choose either C++ or Fortran or Python. We recommend C++ or Fortran. Finally, we recommend that you work together. Optimal working groups consist of 2-3 students, but more people can collaborate. You can then hand in a common report.

Literature

1. B. L. Hammond, W. A. Lester and P. J. Reynolds, Monte Carlo methods in Ab Inition Quantum Chemistry, World Scientific, Singapore, 1994, chapters 2-5 and appendix B.
2. B.H. Bransden and C.J. Joachain, Physics of Atoms and molecules, Longman, 1986. Chapters 6, 7 and 9.
3. S.A. Alexander and R.L. Coldwell, Int. Journal of Quantum Chemistry, **63** (1997) 1001. This article is available at the webpage of the course as the file jastrow.pdf under the project 1 link.
4. C.J. Umrigar, K.G. Wilson and J.W. Wilkins, Phys. Rev. Lett. **60** (1988) 1719.

Unit tests, how and why?

Unit Testing is the practice of testing the smallest testable parts, called units, of an application individually and independently to determine if they behave exactly as expected. Unit tests (short code fragments) are usually written such that they can be preformed at any time during the development to continually verify the behavior of the code. In this way, possible bugs will be identified early in the development cycle, making the debugging at later stage much easier. There are many benefits associated with Unit Testing, such as

- It increases confidence in changing and maintaining code. Big changes can be made to the code quickly, since the tests will ensure that everything still is working properly.
- Since the code needs to be modular to make Unit Testing possible, the code will be easier to reuse. This improves the code design.
- Debugging is easier, since when a test fails, only the latest changes need to be debugged.
- Different parts of a project can be tested without the need to wait for the other parts to be available.
- A unit test can serve as a documentation on the functionality of a unit of the code.

Here follows a simple example, see the website of the course for more information on how to install unit test libraries.

```
#include <unittest++/UnitTest++.h>

class MyMultiplyClass{
public:
   double multiply(double x, double y) {
      return x * y;
   }
};
TEST(MyMath) {
     MyMultiplyClass my; CHECK_EQUAL(56, my.multiply(7,8));
}
int main()
{
return UnitTest::RunAllTests();
}
```

For Fortran users, the link at http://sourceforge.net/projects/fortranxunit/ contains a similar software for unit testing.