Project 3, deadline October 21 at midnight (11.59pm/23.59)

Computational Physics I FYS3150/FYS4150

Department of Physics, University of Oslo, Norway

Fall semester 2019

Project 3, numerical integration, deadline October 21

The task of this project is to integrate first in a brute force manner a six-dimensional integral which is used to determine the ground state correlation energy between two electrons in a helium atom. The integral appears in many quantum mechanical applications. However, if you are not too familiar with quantum mechanics, you can simply look at the mathematical details. We will employ both Gauss-Legendre and Gauss-Laguerre quadrature and Monte-Carlo integration. Furthermore, you will need to parallelize your codes.

We assume that the wave function of each electron can be modelled like the single-particle wave function of an electron in the hydrogen atom. The single-particle wave function for an electron i in the $1s$ state is given in terms of a dimensionless variable (the wave function is not properly normalized)

$$\mathbf{r}_i = x_i\mathbf{e}_x + y_i\mathbf{e}_y + z_i\mathbf{e}_z,$$

as

$$\psi_{1s}(\mathbf{r}_i) = e^{-\alpha r_i},$$

where α is a parameter and

$$r_i = \sqrt{x_i^2 + y_i^2 + z_i^2}.$$

We will fix $\alpha = 2$, which should correspond to the charge of the helium atom $Z = 2$.

The ansatz for the wave function for two electrons is then given by the product of two so-called $1s$ wave functions as

$$\Psi(\mathbf{r}_1, \mathbf{r}_2) = e^{-\alpha(r_1+r_2)}.$$

Note that it is not possible to find a closed-form or analytical solution to Schrödinger's equation for two interacting electrons in the helium atom.

The integral we need to solve is the quantum mechanical expectation value of the correlation energy between two electrons which repel each other via the classical Coulomb interaction, namely

$$\langle \frac{1}{|\mathbf{r}_1-\mathbf{r}_2|} \rangle = \int_{-\infty}^{\infty} d\mathbf{r}_1 d\mathbf{r}_2 e^{-2\alpha(r_1+r_2)} \frac{1}{|\mathbf{r}_1-\mathbf{r}_2|}. \tag{1}$$

Note that our wave function is not normalized. There is a normalization factor missing, but for this project we don't need to worry about that.

This integral can be solved in closed form and the answer is $5\pi^2/16^2$. Can you derive this value?

For this project you can hand in collaborative reports and programs.

Project 3a): Gauss-Legendre Quadrature. Use Gauss-Legendre quadrature and compute the integral by integrating for each variable $x_1, y_1, z_1, x_2, y_2, z_2$ from $-\infty$ to ∞. How many mesh points do you need before the results converges at the level of the third leading digit? Hint: the single-particle wave function $e^{-\alpha r_i}$ is more or less zero at $r_i \approx \lambda$ (find the appropriate limit). You can therefore replace the integration limits $-\infty$ and ∞ with $-\lambda$ and $+\lambda$, respectively. You need to check that this approximation is satisfactory, that is, make a plot of the function and check if the abovementioned limits are appropriate. You need also to account for the potential problems which may arise when $|\mathbf{r}_1-\mathbf{r}_2| = 0$.

Project 3b): Improved Gauss-Quadrature. The Legendre polynomials are defined for $x \in [-1, 1]$. The previous exercise gave a very unsatisfactory ad hoc procedure. We wish to improve our results. It can therefore be useful to change to another coordinate frame and employ the Laguerre polynomials. The Laguerre polynomials are defined for $x \in [0, \infty)$ and if we change to spherical coordinates

$$d\mathbf{r}_1 d\mathbf{r}_2 = r_1^2 dr_1 r_2^2 dr_2 dcos(\theta_1) dcos(\theta_2) d\phi_1 d\phi_2,$$

with

$$\frac{1}{r_{12}} = \frac{1}{\sqrt{r_1^2+r_2^2-2r_1r_2cos(\beta)}}$$

and

$$cos(\beta) = cos(\theta_1)cos(\theta_2) + sin(\theta_1)sin(\theta_2)cos(\phi_1-\phi_2))$$

we can rewrite the above integral with different integration limits. Find these limits and replace the Gauss-Legendre approach in a) with Laguerre polynomials. The function gauss-laguerre.cpp in the CodeExamples folded can be used. Do your results improve? Compare with the results from a).

Important notice for c++ programmers: the function which computes the Gauss-Laguerre integration points and weights returns arrays which start at

1 and end n instead of the default values 0 and $n-1$. You need to declare an array of length $n+1$.

Project 3c): Monte Carlo Integration. Compute the same integral but now with brute force Monte Carlo and compare your results with those from the previous points. Discuss the differences. With bruce force we mean that you should use the uniform distribution.

Project 3d): Improved Monte Carlo Integration. Improve your brute force Monte Carlo calculation by using importance sampling. Hint: use the exponential distribution and transform to spherical coordinates. Does the variance decrease? Does the CPU time used compared with the brute force Monte Carlo decrease in order to achieve the same accuracy? Comment your results and make a list over the time each method uses. Compare the results also.

Project 3e): Improved Monte Carlo Integration and Parallization. Finally, for the last exercise you should parallelize your code using openMP or MPI. Time your program with various compiler flags and comment these results as well. In particular, we want to see whether you achieve an optimal speed-up or not.

Introduction to numerical projects

Here follows a brief recipe and recommendation on how to write a report for each project.

- Give a short description of the nature of the problem and the eventual numerical methods you have used.
- Describe the algorithm you have used and/or developed. Here you may find it convenient to use pseudocoding. In many cases you can describe the algorithm in the program itself.
- Include the source code of your program. Comment your program properly.
- If possible, try to find analytic solutions, or known limits in order to test your program when developing the code.
- Include your results either in figure form or in a table. Remember to label your results. All tables and figures should have relevant captions and labels on the axes.
- Try to evaluate the reliabilty and numerical stability/precision of your results. If possible, include a qualitative and/or quantitative discussion of the numerical stability, eventual loss of precision etc.
- Try to give an interpretation of you results in your answers to the problems.

- Critique: if possible include your comments and reflections about the exercise, whether you felt you learnt something, ideas for improvements and other thoughts you've made when solving the exercise. We wish to keep this course at the interactive level and your comments can help us improve it.

- Try to establish a practice where you log your work at the computerlab. You may find such a logbook very handy at later stages in your work, especially when you don't properly remember what a previous test version of your program did. Here you could also record the time spent on solving the exercise, various algorithms you may have tested or other topics which you feel worthy of mentioning.

Format for electronic delivery of report and programs

The preferred format for the report is a PDF file. You can also use DOC or postscript formats or as an ipython notebook file. As programming language we prefer that you choose between C/C++, Fortran2008 or Python. The following prescription should be followed when preparing the report:

- Use Devilry to hand in your projects, log in at `http://devilry.ifi.uio.no` with your normal UiO username and password and choose either 'fys3150' or 'fys4150'. There you can load up the files within the deadline.

- Upload **only** the report file! For the source code file(s) you have developed please provide us with your link to your github domain. The report file should include all of your discussions and a list of the codes you have developed. Do not include library files which are available at the course homepage, unless you have made specific changes to them.

- In your git repository, please include a folder which contains selected results. These can be in the form of output from your code for a selected set of runs and input parametxers.

- In this and all later projects, you should include tests (for example unit tests) of your code(s).

- Comments from us on your projects, approval or not, corrections to be made etc can be found under your Devilry domain and are only visible to you and the teachers of the course.

Finally, we encourage you to work two and two together. Optimal working groups consist of 2-3 students. You can then hand in a common report.