FYS3150: Project 3: Numerical Integration

Henrik Haugerud Carlsen, Martin Moen Carstensen and Øyvind Augdal Fløvig

October 21, 2019

Abstract

In this report we have take a look at Gauss-Legendre and Gauss-Laguerre Quadrature as well as Monte Carlo integration for solving integrals. The problem more specifically is to determine the expectation value of the correlation energy between two electrons by solving an integral. Paralellization is a way of optimizing runtime of a program. A speedup of 2.2 was reached on a computer with 2 processors, and a speedup of 4 was reached on a computer with 4 processors. For a large number of computations parallelization greatly decreases runtime, but for a small number of computations parallelization might increase runtime.

1 Introduction

This report is going to look at different ways of integrating the same integral which is the expectation value of the Coulomb interaction. This will be done by using Gaussian quadrature method with Legendre polynomials for the Cartesian coordinates and a mix of both Legendre and Laguerre polynomials for when we switch to spherical coordinates. Furthermore we want to use Monte Carlo Integration as well to calculate the integral. This is done so the different methods can be compered in terms of speed, accuracy. This is done to see how different integration methods interact differently and what the benefits of using them are. Lastly parallelization of the Monte Carlo method will be looked at for, potentially, even greater time reduction.

2 Method

This report is going to look at different numerical methods of integrating the same integral. We are going to consider the six-dimensional integral which is used to determine the ground state correlation energy between to electrons in a helium atom. The wave function of one electron in the 1s

state with position

$$\vec{r}_i = x_i\hat{e}_x + y_i\hat{e}_y + z_i\hat{e}_z$$

is

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

Here α is a parameter which we are going to fix to $\alpha = 2$ to model the helium atom of charge $Z = 2$ and

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

The anzats of two electron is then given as the product of two single electron 1s states which is

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

It is not possible to find a closed-form or analytical solution of this Schrodinger's equation for two interacting electrons in the Helium atom. Instead we can find the quantum mechanical expectation value of the correlation energy between the two electrons. This can be done with the equation

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

2.1 Gaussian Quadrature

Any integral of the form

$$I = \int f(x)dx = \int W(x)g(x)dx$$

can be written as a sum of the wieghts and function values at the form

$$I = \int f(x)dx = \sum w_i g(x_i)$$

Here the weights and mesh points are found using orthogonal polynomials of degree N. And these points can determine the integral value of a polynomial of degree 2N-1 exact. This is being done with two different polynomials of Legendre and Laguerre. Legendre polynomials are defined on the interval from -1 to 1 so this approximation works well on integrals with finite limits, while the Laguerre polynomials are determined on the interval from 0 to ∞ which is useful when considering sphercial coordinates

2.1.1 Gauss-Legendre

With the Gauss-Legendre method we first find the N roots of the Legendre polynomial which are going to be the x_i mesh points. From these mesh

points we can find the corresponding weights with the recurrsion formula where

$$L_{j+1}(x_i) = ((2j+1)x_i L_j(x_i) - jL_{j-1}(x_i))/(j+1)$$

From this we can find the derivated of the Nth polynomial with the formula

$$L'_N(x_i) = N\frac{(x_i L_N(x_i) - L_{N-1})}{x_i^2 - 1}$$

Now the weights are found by using

$$\omega_i = \frac{2}{(1-x_i^2)L'_n(x_i)^2}$$

From this we can now integrate any function defined on the interval from -1 to 1 with the weights ω_i and mesh points x_i. However if we want to do this for a more general integral that goes from a starting point a to an end point b we need to do a change of variables.

$$I = \int_a^b f(t)dt = \frac{b-a}{2}\int_0^1 f\left(\frac{(b-a)x}{2} + \frac{(b+a)}{2}\right)dx$$

so now we have everything we need to compute a general integral as

$$I = \int_a^b f(t)dt \approx \frac{b-a}{2}\sum_i^N \omega_i f\left(\frac{(b-a)x_i}{2} + \frac{(b+a)}{2}\right)$$

By looking at our original integral in Cartesian coordinates we get the form

$$\int e^{-2\alpha(\sqrt{x_1^2+y_1^2+z_1^2}+\sqrt{x_2^2+y_2^2+z_2^2})}\frac{dx_1dx_2dy_1dy_2dz_1dz_2}{\sqrt{(x_1-x_2)^2+(y_1-y_2)^2+(z_1-z_2)^2}}$$

By then setting the limits at $b = -a = \lambda$, where $\lambda = 3$ we can use a six dimensional Gauss-Legendre integral to calculate our original integral

2.1.2 Gauss-Laguerre

Similarly to the Gauss-Legendre method we find the new mesh points by the roots of the N-th Laguerre polynomial. Furthermore the weights are found with the same Laguerre polynomial which can be found with the recursion formula of

$$\mathcal{L}_{j+1}(x_i) = ((2j+1-x_i)\mathcal{L}_j(x_i) - j\mathcal{L}_{j-1}(x_i))/(j+1)$$

The weights of the N-th Laguerre polynomials are now given as

$$\omega_i = \frac{x_i}{(N+1)^2 \mathcal{L}_{N+1}(x_i)^2}$$

So now we can calculate an integral of the form

$$I = \int_0^\infty e^{-x} f(x) dx$$

where e^{-x} is the weight function as a sum of

$$\sum_i \omega_i f(x_i) dx$$

By now changing the integral to spherical coordinates we get for

$$d\vec{r}_1 d\vec{r}_2 = r_1^2 r_2^2 sin\theta_1 sin\theta_1 dr_1 dr_2 d\theta_1 d\theta_2 d\phi_1 d\phi_2$$

with

$$\frac{1}{|\vec{r}_1 - \vec{r}_2|} = \frac{1}{\sqrt{r_1^2 + r_2^2 - 2r_1 r_2 cos(\beta)}}$$

where

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

This gives that the total integral becomes

$$I = \int_0^\infty \int_0^\infty \int_0^\pi \int_0^\pi \int_0^{2\pi} \int_0^{2\pi} e^{-2\alpha(r_1+r_2)} \frac{r_1^2 r_2^2 sin\theta_1 sin\theta_1 dr_1 dr_2 d\theta_1 d\theta_2 d\phi_1 d\phi_2}{\sqrt{r_1^2 + r_2^2 - 2r_1 r_2 cos(\beta)}}$$

So here we can use Gauss-Legendre method for integrating the angles and Gauss-Legendre for integrating the radial part. Here one of the exponential powers are used as the weight function as $W(x) = e^{-(r_1+r_2)}$

2.2 Monte Carlo Integration

2.2.1 Brute force approach

2.2.2 Exponential distribution approach

2.3 Paralellization

Parallelization is a powerful tool to make a programs run faster. The idea is to distribute the work/computations in a program among available processors. Instead of one processor running through the program sequentially parts of the program now gets computed simultaneously saving time. As

an example, for loops can be parallized. The aim would be to distribute the computations among the available cores, do the computations, and then gather the computed values.To parallelize "SmarterMonteCarlo" a function "main()" containing the following is used:

```
pool = cpucount()                    ▷ "pool" counts the available cpus.
result = pool.map(MonteCarlo, list)  ▷ pool.map distributes the
computations in the function "MonteCarlo" as function of "list".
```

The idea is to find out how many cpus are available, this is done by the function "cpucount()". The parallelization of the function "MonteCarlo" is initiated in the variable "result" where the "map"-function distributes the computations done in "MonteCarlo" for all the elements in the list "list".

3 Results

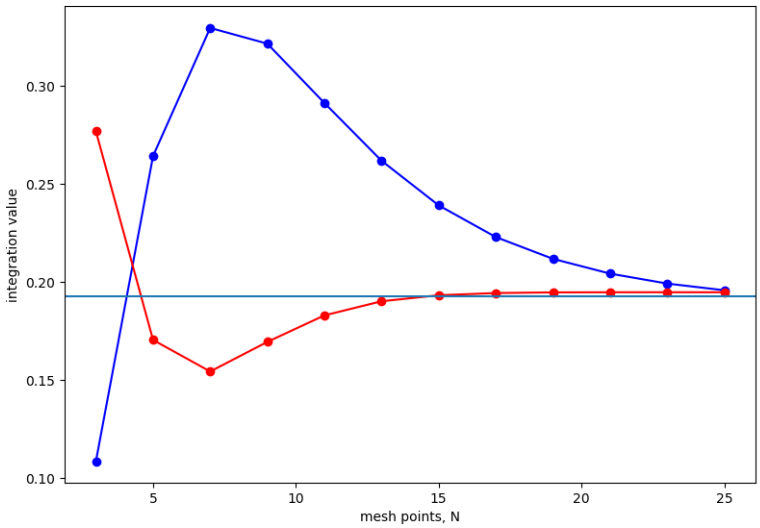

Figure 1: Legendre and Laguerre methods as a function of meshpoints

SETT INN TABELL FOR TIDSBRUK MED 4 KJERNER OG " KJERNER

Table 1: Integration value and time spent for different mesh points with the Gaussian Legendre method

N	Integral value	time spent(s)
3	0.108672	0.0209
5	0.264249	0.4997
7	0.329525	3.0389
9	0.321518	12.4357
11	0.291261	47.9585
13	0.261821	116.628
15	0.239088	265.02
17	0.222933	574.50
19	0.211832	1120.19
21	0.204307	1898.89
23	0.199232	2654.37
25	0.195817	4542.21

Table 2: Integration value and time spent for different mesh points with the Gaussian Laguerre method

N	Integral value	time spent(s)
3	0.276857	0.01396
5	0.170492	0.3132
7	0.154422	2.3425
9	0.169505	10.4558
11	0.183022	35.6265
13	0.190218	95.9022
15	0.193285	235.989
17	0.194396	512.491
19	0.194732	962.119
21	0.194807	1684.94
23	0.194812	3203.78
25	0.194804	4877.90

4 Discussion

4.1 Parallelization

Both the program "SmarterMonteCarlo.py" , and "MonteCarloDone.py" are small programs and does not use much time to run. None of them include for loops and are both vectorized. Parallelization is useful if a program has multiple function calls and the calls are independent of eachother. The process of scattering and gathering data would in this case not be useful because of the low runtime of the programs, it might even make the pro-

grams slower. An argument can be made for not parallizing a program if it is contemptuously fast because the time it takes to distribute computations and gather them again is sometime larger than the time saved by parallizing. The amount of available processors is, as expected, crutial for the times saved by parallizing. For a pc with two cores the the speedup of "Parallel-MonteCarlo.py" was 2.2, and for a computer with four cores the speed up was 4.