FYS4150-project 3
Numerical integation

Morten Hemmingsen

October 21, 2019

Github adress for this project: `https://github.com/mortenhem/FYS-4150-`

Abstract

We solve a six dimensional integral by Gauss quadrature and Monte Carlo integration. The Gauss-Quadrature methods are very slow and not very reliable for higher dimension integrals.

1 Introduction

The integral in question is

$$\int_{-\infty}^{\infty} d\mathbf{r_1} d\mathbf{r_2} e^{-2\alpha(r_1+r_2)} \frac{1}{|\vec{r_1} - \vec{r_2}|}$$

Where

$$dr_i = dx_i dy_i dz_i$$

in cartesian coordiantes. We will solve this integral with Gauss-Legendre Quadrature, Gauss-Laguerre Quadrature and Monte-Carlo integration.

2 Method

2.1 Gauss-Legendre Quadrature

The first method used is Gauss-Legendre Quadrature. The idea is to approximate the integrand by orthogonal polynomials by extracting a weight function. We start by representing the integrand by a polynomial of degree $2N-1$,

$$f(x) \approx P_{2N-1}(x)$$

$$I = \int_a^b f(x)dx = \int_a^b P_{2N-1}(x) = \sum_{i=0}^{N-1} P_{2N-1}(x_i)\omega_i$$

, where x_i are the zeros of the orthogonal polynomial of degree N, and ω_i are the weights.

The weights are given by $2(L^{-1})_{0i}$ with L being the matrix.

$$L = \begin{bmatrix} L_0(x_0) & L_1(x_0) & \cdots & L_{N-1}(x_0) \\ L_0(x_1) & L_1(x_1) & \cdots & L_{N-1}(x_1) \\ \cdots & \cdots & \cdots & \cdots \\ L_0(x_{N-1}) & L_1(x_{N-1}) & \cdots & L_{N-1}(x_0) \end{bmatrix}$$

Integrals can be solved by Gauss-Legendre quadrature if the integration interval is (-1,1), but in fact, all intervals are permitted by only doing a change of variables

$$x = \frac{(b-a)}{2}t + \frac{b+a}{2}$$

where a and b is the lower and upper integration limit, respectively.

We use Gauss-Legendre quadrature to solve the six dimensional intetgral in cartesian coordinates. The integral reads

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \frac{e^{-2\alpha(\sqrt{x_1^2+y_1^2+z_1^2}+\sqrt{x_2^2+y_2^2+z_2^2})}}{\sqrt{(x_1-x_2)^2+(y_1-y_2)^2+(z_1-z_2)^2}} dx_1 dx_2 dy_1 dy_2 dz_1 dz_2$$

The integration limits are infinite but if we make a plot of the exponential function, we see that it is very close to zero for $r_i = 4$. We can therefor integrate over the interval (-4,4) instead of $(-\infty, \infty)$

2.2 Gauss-Laguerre Quadrature

The next method of integration is the Gauss-Laguerre quadrature. The weight function is now $W(x) = x^{\alpha}e^{-x}$. We will use Laguerre polynomials which are orthogonal and defined for x in (0,∞) The procedure is the same as for the Gauss-Legendre Quadrature. I chose to compute the mesh points and weights by using the numpy module polynomial.laguerre.laggauss.

To compute the integral by this method we rewrite the integral using spherical coordinates:

$$d\vec{r_1}d\vec{r_2} = r_1^2 sin(\theta_1) r_2^2 sin(\theta_2) dr_1 dr_2 d\theta_1 d\theta_2 d\phi_1 d\phi_2$$

. The integral becomes

$$\int_0^{2\pi}\int_0^{2\pi}\int_0^{\pi}\int_0^{\pi}\int_0^{\infty}\int_0^{\infty}\frac{e^{-2\alpha(r_1+r_2)}}{r_{12}} r_1^2 sin(\theta_1) r_2^2 sin(\theta_2) dr_1 dr_2 d\theta_1 d\theta_2 d\phi_1 d\phi_2$$

where $r_{12} = \sqrt{r_1^2 + r_2^2 - 2r_1 r_2 cos(\beta)}$ and

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

2.3 Monte Carlo Integration

The Monte Carlo method works by writing the integral as an average value for a given probability distribution function.

$I =< f >= \int_a^b f(x)p(x)dx$. When using the uniform distribution, $p(x) = 1$ we get

$$I = \int_0^1 f(x)dx$$

. The integral is then the average over the interval [0,1]

The program for Monte Carlo integration is not finished at this stage. No results are therefore produced.

3 Results

N	Legendre	Error	CPU-time[sec]
2	0.00000	0.19276	0.005
5	0.12220	0.070557	0.332
10	0.03125	0.16151	9.774
15	0.28715	0.09438	111.726
20	0.12751	0.06525	594.390
25	0.21336	0.02059	2262.664
30	0.16374	0.02902	6879.878

The results from the Gauss-Legendre method is not very good. Even for a small number of points, the time spent is way to high, and the results are not good enough considering the time spent.

N	Laguerre	Error	CPU-time[sec]
2	0.10436	0.0883	0.0007
5	0.60387	0.04111	0.1602
10	1.5626	1.3699	10.022
15	2.5823	2.3896	113.822
20	3.6247	3.4320	636.684
25	4.6778	4.4853	2455.078
30	5.7378	5.544	7262.98

The results from the Gauss-Laguerre method is not very good. This is probably due to some error in the program that is yet to be discovered. But the time spent is equally horrifying to the Gauss-Legendre method.

4 Remarks

Only the Gauss-Legendre quadrature method has produced results which seems to be somewhat correct. The Gauss-Laguerre method is probably not implemented correctly due to the fact that the results deviate greatly from the correct value. The methods require to much time to run additional trys at this stage.