Computational Physics - Project 3

Johannes Scheller, Vincent Noculak, Lukas Powalla

October 16, 2015

Contents

1 Introduction to Project 3

2 Theoretical background for numerical integration

2.1 Gaussian quadrature

2.2 Montecarlo integration

3 Execution

3.1 Gaussian quadrature

3.2 Montecarlo integration

We calculated the same integral with montecarlo method. First, we calculated the integral in a brute force way. This means that we calculate the integral using (pseudo) random numbers, which obey uniform distribution functions. The random numbers for each of the six dimensional integral are uniform distributed in a chosen interval (-a to a). Furthermore, we don't transform the integral, but we calculate it in Cartesian coordinates. We got the values in table 1. (We used the interval for a=2) In one dimension, the montecarlo method can be described by formula 1.

$$I = \int_a^b f(x)dx \approx < f(x) > \cdot(b-a) = \frac{1}{n}\sum_{i=1}^{n} f(x_i)\cdot(b-a) = \hat{I} \tag{1}$$

In addition to that, we tried to improve our calculations. First, we transformed the integral into spherical coordinates. In spherical coordinates, we use the variables θ (from 0 to π), ϕ (from 0 to 2π) and r (from 0 to infinity) instead of using Cartesian coordinates $x_{i,k}$ [$-\infty$ t ∞) (i=1,2,3; k=1,2). We also used a distribution function in order to get appropriate values of the random numbers. Formula 2 to 4 describe the general one dimensional reformulation if you want to use a other particle distribution function.

$$P(x) = \int_0^x p(x)dx \tag{2}$$

$$I = \int_a^b \frac{f(x)}{p(x)}\cdot p(x)dx = \cdot\int_a^b \hat{f}(x)\cdot p(x)dx \approx \frac{1}{n}\sum_{i=1}^{n}\hat{f}(y_i)\cdot(b-a) = \hat{I} \tag{3}$$

$$y_i(x_i) = P^{-1}\left(p(y(x_i))\right) = P^{-1}(x_i) \tag{4}$$

In order to improve the precision of the integral, we used a not uniform distribution function, which can be found in formula 5ff.

$$P(x) = \int_0^x 4\cdot e^{-4x} = 1 - e^{-4x} \tag{5}$$

$$y_i(x_i) = -\frac{1}{4}ln(1-x_i) \tag{6}$$

We transformate the integral to spherical Coordinates and use the distribution function for r_1 and r_2. Finally, the integral can be calculated through formula 8. The results are in table 2.

$$f(r_{1,i}, r_{2,i}, \theta_{1,i}, \theta_{2,i}, \phi_{1,i}, \phi_{2,i}) = \frac{r_{1,i}^2\cdot r_{2,i}^2\cdot sin(\theta_{1,i})sin(\theta_{2,i})}{\sqrt{r_{1,i}^2 + r_{2,i}^2 - 2\cdot r_{1,i}r_{2,i}cos(\theta_{1,i})cos(\theta_{2,i}) + sin(\theta_{1,i})sin(\theta_{2,i})\cdot cos(\phi_{1,i}-\phi_{2,i})}\cdot 4^2} \tag{7}$$

$$\hat{I} = \frac{1}{n}\sum_{i=1}^{n} f(r_{1,i}, r_{2,i}, \theta_{1,i}, \theta_{2,i}, \phi_{1,i}, \phi_{2,i})\cdot(2\pi-0)^2\cdot(\pi-0)^2 \tag{8}$$

4 Comparison and discussion of the results

5 Source-code

Table 1: Data from the brute force montecarlo algorithm (part c))

n	Integral	standart deviation	time in s
100	0.0563094	0.0362644	0
1000	0.226437	0.159513	0.001
10000	0.0986332	0.0263534	0.006
100000	0.155652	0.01757	0.062
1000000	0.178452	0.00754909	0.667
10000000	0.188683	0.00290854	6.645
100000000	0.19169	0.000942637	70.881

Table 2: Data from the montecarlo algorithm with distribution function (in spherical coordinates) (part d))

n	Integral	standart deviation	time in s
100	0.216734	0.0769193	0
1000	0.173694	0.0241284	0.002
10000	0.186069	0.00885492	0.021
100000	0.19571	0.00343622	0.21
1000000	0.193259	0.00101078	2.07