Numerical Integration

Numerical Introductory Course

Marvin Gauer (580553)

Contents

1. Motivation

Integration is an important operation in mathematics. Unfortunately, in real life applications one might find it extremely difficult or even impossible to solve certain integrals in a closed form. Due to the continous improvement in computational power one might address this issue by numercially approximating the integral of interest. In order to do so, several procedures have been developed, each with it's own advantages respectively disadvantages. In this document we want to present 5 methods for numerical integration and furthermore apply one of those methods to numerically integrate the normal distributions density in order to approximate the normals cumulative distribution function.

2. Literature Review

lalalala

3. Theory

In the following section I want to explain some of the most popular methods in numerical integration. These can be distingushed into one and multi-dimensional methods. Furthermore one might distinguesh numerical integration methods further into deterministic and probabilistic methods. But before I start introducing the methods of interest I will do a little recap of the bascis:

3.1 Review: The Riemann Integral

The Riemann Integral is one of the two classic concepts of integrals in analysis. It is named after the German mathematician Bernhard Riemann and it's aim is to calculate the area between the x-axis and a certain limited function $f : [a; b] \to \mathbb{R}$. Loosley speaking, the basic idea behind the concept is to approximate the desired integral by summing up different areas of easier to compute rectangles.

The kind of definition I want to present here is the definition using upper and lower sums introduced by Jean Gaston Darboux:

Let $f : [a; b] \to \mathbb{R}$ be a limited function and $[a; b]$ be an interval. Furthermore, let P be a partition of $[a; b]$ where $a = x_0 < x_1 < ... < x_{n-1} < x_n = b$. Then we can define the upper and lower sums accordingly:

$$U(P) = \sum_{k=1}^{n}((x_k - x_{k-1}) \cdot \sup_{x_{k-1}<x<x_k} f(x))$$

$$L(P) = \sum_{k=1}^{n}((x_k - x_{k-1}) \cdot \inf_{x_{k-1}<x<x_k} f(x))$$

Now we can compute the infimum and supremum of the upper and lower sum over all partitions P. Therefore it follows:

$$\sup_P L(P) \leq \inf_P U(P)$$

In case of equality, on says that f in Riemann integrable.

3.2 One-Dimensional Procedures

The one dimensional procedurees elaborated on in this chapter are classified as deterministic methods. Throughout this document the function $f : [-4; 4] \to \mathbb{R}$ with $f(x) = x^2 + 3 \cdot x + 4$ is used for visualizing the procedures introduced.

3.2.1 Midpoint Quadrature

The idea of the Midpoint Quadrature directly derives from the definition of the Riemann Integral. We therefore want to calculate the area between the x-axis and a limited function $f : [a; b] \to \mathbb{R}$. The algorithm works in the way, that we start by partitioning our interval of interest $[a; b]$ into aquidistant subintervals $a = x_0 < x_1 < ... < x_{n-1} < x_n = b$ with stepwidth $h = \frac{b-a}{n}$. Afterwards we calculate the midpoint $x^{(i)}$ within each subinterval $[x_i; x_{i+1}]$ for $i \in \{0, 1, ..., n-1\}$ and evaluate f for each $x^{(i)}$. For our approximation it then holds that $\int_a^b f(x)dx \approx \sum_{k=0}^{n-1} f(x^{(i)} \cdot (x_{i+1} - x_i)$.

An illustration of the procedure can be found in Figure 1.

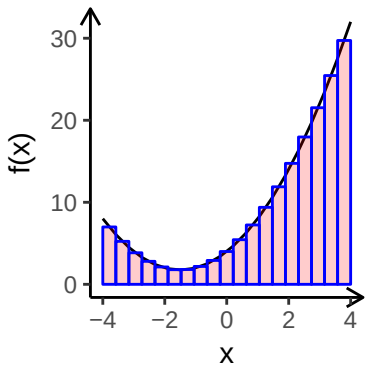

Figure 1: Illustration of the Midpoint or Rectangular Quadrature

When numerically solving an integral one is naturally interested in the error of the approximation which will in the following be denoted by $E(f)$. In case of f having a continous

second derivative on $[a;b]$ meaning $f \in C^{(2)}_{[a;b]}$, an upper bound for the error of the Midpoint quadrature can be specified as follows:

$$E(f) = \frac{(b-a)^3}{24 \cdot n^2} \cdot \max_{a \leq x \leq b} |f^{“}(x)|$$

3.2.2 Simpson-Rule

The Simposn-Rule is similiar to the Rectangular Quadrature, but instead of rectangles quadratic functions are used in order to calculate the area between the x-axis and our limited function $f : [a;b] \to \mathbb{R}$ more accurately. We again start by partitioning our interval of interest $[a;b]$ into n subintervals $a = x_0 < x_1 < ... < x_{n-1} < x_n = b$ with equidistant distances which we will in the following denote by $\Delta x = \frac{b-a}{n}$. Afterwards we calculate the midpoint $x^{(i)}$ within each subinterval $[x_i; x_{i+1}]$ for $i \in \{0, 1, ..., n-1\}$. Now we use the 3 points $(x_i; f(x_i))$, $\left(x^{(i)}; f(x^{(i)})\right)$ and $(x_{i+1}; f(x_{i+1}))$ within each subinterval to interpolate our quadratic functions $g_i(x) : [x_i; x_{i+1}] \to \mathbb{R}$. For our approximation it then holds that $\int_a^b f(x)dx \approx \frac{\Delta x}{6} \cdot \left(f(x_0) + 2 \cdot \sum_{k=1}^{n-1} f(x_k) + f(x_n) + 4 \cdot \sum_{k=0}^{n-1} f(x^{(k)})\right)$.

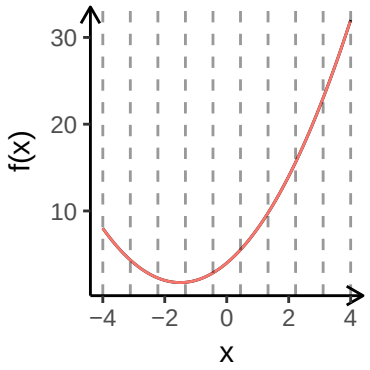

Figure 2: Illustration of the Simpson Rule

The Simpson's rule is an approximation. As with any approximation, before you can safely use it, you must know how good (or bad) the approximation might be.The error of this approximation will in the following be denoted by $E(f)$. In case of f having a continous fourth derivative on $[a;b]$ meaning $f \in C^{(4)}_{[a;b]}$, an upper bound for the error of the Simpson's quadrature can be specified as follows:

$$E(f) \leq= \frac{(b-a)^5}{2880 \cdot n^4} \cdot \max_{a \leq x \leq b} |f^{(4)}(x)|$$

3.2.3 Adaptive Algorithm

When calculating integrals numerically one might in some cases not have unlimited computational power. In this case it might not be smart to use equidstant stepwidth in your numerical

integration method. In this case it might be clever to use a wider stepwidth in an area where we can approximate our function well and a narrower one where our function cannot be approximated that well. The adaptive algorithm solves exactly that issue by minimizing the local error (the error within each subinterval) until it reaches a certain error threshold. In case of a "well behaved" functions the algorithm works just as fine as the algorithms above but outperforms when we want to numerically integrate "badly behaved" functions. The algorithm therefore starts with an initial number of subintervals m. It then approxima

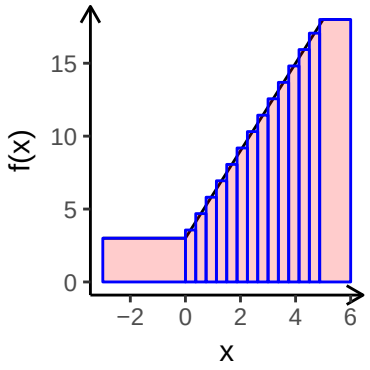

Figure 3: Illustration of the adaptive Midpoint Quadrature

3.3 Multi-Dimensional Procedures

In lots of applications multidomensional Inetgrals need to be solved numerically. This means that we now want to calculate the integral $\int_\Omega f(\mathbf{x})d\mathbf{x} = \int_{a_1}^{b_1} ... \int_{a_m}^{b_m} f(x_1, x_2, ..., x_m)dx_1dx_2...dx_m$. One might have the idea, to use the quadrature rules explained above in a multidimensional setting to calculate the integral at hand, but we can easily see that this will end in the **Curse of dimensionality**. Therefore I want to concentrate on Monte Carlo integration methods going on. Please note that the goodness of your approximation depends on your random number generator when applying Monte Carlo integration methods.

3.3.1 Crude

The Crude Monte Carlo Integration method is a fairly simple method to calculate integrals numerically. It works in the way, that we partition k bins $[a_{1,j}; b_{1,j}] \times [a_{2,j} \times b_{2,j}] \times ... \times [a_{m,j}; b_{m,j}]$ $\forall j \in \{1, ..., k\}$. Furthermore, one needs to generate n m-dimensional uniformally distributed points $\mathbf{x_i} = (x_1, ..., x_m)$ $\forall i \in \{1, ..., n\}$. In the next step, we calculate $f(\mathbf{x_i})$ $\forall i \in \{1, ..., n\}$ and the means m_i for each bin. Therefore it holds that $\int_a^b f(x)dx \approx \sum_{t=1}^{k} (\prod_{l=1}^{m} (b_{l,t} - a_{l,t}) \cdot m_t)$

3.3.2 Hit or Miss

On the other hand, the Hit or Miss Monte Carlo Integration method works slidly differently. Here we create n $m+1$-dimensional uniformly distributed points $(x_{i,1}, ..., x_{i,m}, y_i)$ for $i \in$

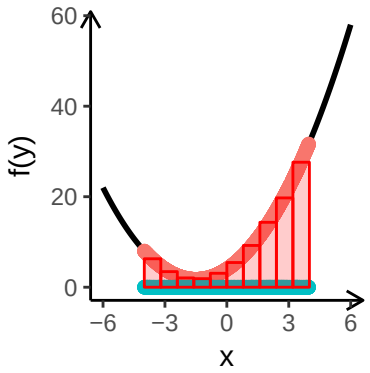

Figure 4: Illustration of the Crude Monte Carlo integration method

$\{1, .., n\}$. The first m dimensions are for our domain $[a_1; b_1] \times [a_2 \times b_2] \times ... \times [a_m; b_m]$ and are always uniformaly distributed on $[a_i; b_i]$ $\forall i \in \{1, ..., m\}$. The y_i coordinate is for our $f(\mathbf{x})$-values and these are uniformally distributed on $[0; max(f(\mathbf{x}))]$. Now the percentage of points p for which holds $y_i \leq f(x_1, ..., x_m)$ and the total area/vloume A surrounding the graph needs to be calculated. A is basically the area/volume of the domain of the generated random points. Therefore it holds that $\int_a^b f(x)dx \approx A \cdot p$

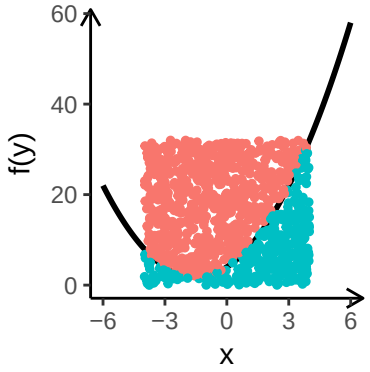

Figure 5: Illustration of the Hit or Miss Monte Carlo integration method

Obviously, the quality of the answer depends on the quality of the random number generator sequence which is used.

3.4 Integrals over infinite Intervals

Working with the procedures elaborated on above one needs to have certain limits for the integral to approximate, but in many applications one might also be interested in numerically solving improper integrals. I therefore want to elaborate on a method known as change of variables in order to account for the issue described above. The basic idea of the procedure is to adjust the integrand in such a way, that we have an equivalent definite integral at the end. Therefore we can use the following three functions to trasform our integral at hand:

$$\int_{-\infty}^{\infty} f(x)dx = \int_{-1}^{1} f(\frac{t}{1-t}) \cdot \frac{1+t^2}{(1-t^2)^2} dt$$
$$\int_{a}^{\infty} f(x)dx = \int_{0}^{1} f(a + \frac{t}{1-t}) \cdot \frac{1}{(1-t)^2} dt$$
$$\int_{-\infty}^{a} f(x)dx = \int_{0}^{1} f(a - \frac{1-t}{t}) \cdot \frac{1}{t^2} dt$$

4. Application: Approximation of the Normal Distribution

Numerical integration is widely used by practitioners. Some of the most well known respectively used applications are among the following:

- Approximation of probabilities of the normal distribution
- Approximation of antiderivatives
- Calculation of moments
- Calculation of the Value at Risk and Expected Shortfall

The application I want to elaborate on is the approximation of the antiderivative of the normal distribution since there exists no closed form solution. Therefore we first need to decide on the sampling points x_i for $i = 1, ..., N$ of our distribution. I choose all integers between -5 and 6. Due to the fact, that the function is quite well behaved there is no need to apply an adaptive algorithm. Therefore I will use the midpoint quadrature applying $15 \cdot i$ bins as a method of choice. Furthermore a change of variables is used in order to account for the improper integral.

To calculate $F(x)$ we now need to solve the following equation for our N sampling points:

$$y_i = F(x_i) \approx \int_{-\infty}^{x_i} f(x)dx \;\; \forall i \in \{1, .., N\}$$

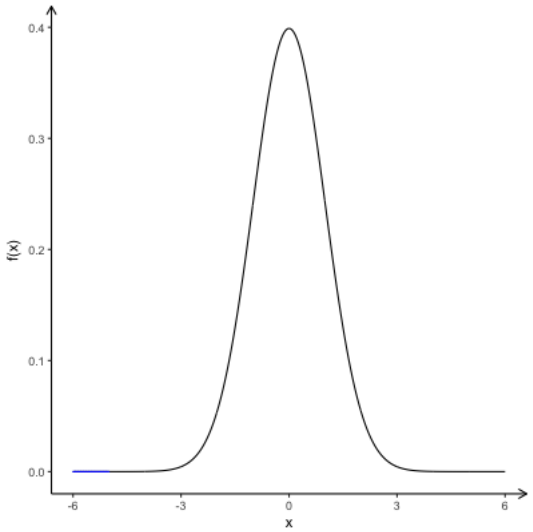

Figure 6: Visualization of the integration

The results of our approximation can be found visualized in Figure 6 and in written form below:

Table 1: Results for the sampling points

x	Number of Bins	Approx. y	y	Error
-5	30	0.00000029	0.00000029	0.00000000
-4	45	0.00003167	0.00003167	-0.00000001
-3	60	0.00134984	0.00134990	-0.00000005
-2	75	0.02275013	0.02275013	0.00000000
-1	90	0.15865653	0.15865525	0.00000127
0	105	0.50000307	0.50000000	0.00000307
1	120	0.84134688	0.84134475	0.00000214
2	135	0.97725037	0.97724987	0.00000050
3	150	0.99865014	0.99865010	0.00000004
4	165	0.99996833	0.99996833	0.00000000
5	180	0.99999971	0.99999971	0.00000000
6	195	1.00000000	1.00000000	0.00000000

It may be noted, that the y is based on R's pnorm()-function. Now that we have our sampling points evaluated we need to decide on a suited function to fit the points. In our case I decided to use a sigmoid function that is fitted using a non-linear least squares approach. This then yields:

$$\hat{F}(x) = \frac{\hat{a}}{1 + \exp^{-\hat{b}\cdot(x-\hat{c})}} = \frac{1}{1 + \exp^{-1.697\cdot(x-0.0044)}}$$

After approximating our function we now need to test the goodness of fit. We will do this by applying a normality test. The normality test of choice is the Shapiro-Wilk test, since it is more powerful than the Anderson-Darling, Lilliefors and Kolmogorov-Smirnov test according to Razali (2011). In order to test whether there is evidence that our approximated distibution is not normal, we need to sample datapoints. For this we create 50 uniformly on $[0; 1]$ distributed random variables. Then we use the inverse of our fitted function to have sample datapoints of corresponding to the approximated CDF $\hat{F}$. Applying the Shapiro-Wilks test to our 50 datapoints yields a p-value of 0.6666. Therefore we cannot reject the null hypothesis, meaning we could not find enough evidence to conclude, that the datapoints are not normally distributed. Furthermore we want to plot the approximated and R's pnorm-function. The plot can be found below and we can easily see, that only slight differences can be spotted indicating, that our approximation already fits the normal distributions CDF quite well.

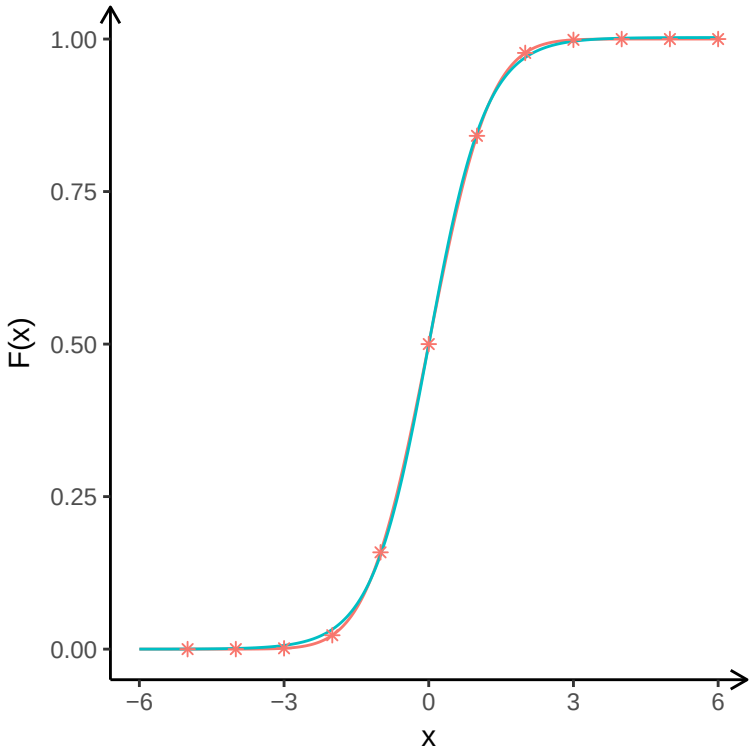

Figure 7: Visualization of the fitted and R's Normal Distributions CDF

5. Conclusion

lalalala

6. Bibliography

("Monte-Carlo-Integration")

"Monte-Carlo-Integration." https://www.mathematik.tu-clausthal.de/interaktiv/integration/montecarlo/.

Razali, Yap Bee, Nornadiah; Wah. 2011. "Power Comparisons of Shapiro–Wilk, Kolmogorov–Smirnov, Lilliefors and Anderson–Darling Tests." *Journal of Statistical Modeling and Analytics* 2 (1): 21–33.