HUMBOLDT-UNIVERSITÄT · ZU BERLIN ·

Numerical Integration

Numerical Introductory Course
Marvin Gauer (580553)

Abstract

Numerical Integration is one of the most widely used numerical methods. It is used by practioners all around the world and in lots of disciplines. The methods might for example be used to . . . .

Due to its relevance, I want to give you a brief overview over some basic numerical integration methods, namely the midpoint rule, simpson rule, crude and hit or miss monte carlo as well as an introduction to the idea of adaptive quadrature. Furthermore, I want to show how one may use numerical integration methods to approximate the cumulative distribution function of a normal distribution. Here we will see, that even tough just basic methods are applied, one can easily approximate the function so well, that some of the most common normality tests may not find enough evidence to reject the null hypothesis of a normal distribution.

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.

2. Literature Review

Most of the sources used for this document are textbooks. This is due to the fact that the majority of procedures described respectivly used in this document are already well established and recently published papers on numerical integration usually deal with very specific applications and are therefore not very helpful in case of an introduction. One of the main sources of this document for example is (Davis J. Philip 2007). The book contains mathematically very well and precisely defined procedures and algorithms. Other well written sources used which explicitly describe Monte Carlo methods are (Gentle 2003) and (Rizzo 2007). Even though numerical integration is not the main topic of these books they still describe monte carlo methods and the application of numerical integration very well.

3. Theory

In the following section I want to explain some of the most well known methods in numerical integration. These can be distinguished into one and multi-dimensional methods. Furthermore one might distinguish 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 Riemann integral, since we are assuming throughout this document, that our functions of interest are Riemann integrable.

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, one says that f in Riemann integrable.

3.2 One-Dimensional Procedures

The one dimensional proceduree elaborated on in this chapter are classified as deterministic methods. Throughout this chapter the function $f : [-4; 4] \to \mathbb{R}$ with $f(x) = x^2 + 3 \cdot x + 4$ is used for visualizing the procedures introduced. The only exception from is the adaptive algorithm, here we use a function different from f described above.

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 equidistant 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.

When numerically solving an integral one is naturally interested in the error of the approximation which will in the following be denoted by $E(f)$. According to (Davis J. Philip 2007, 54), 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:

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

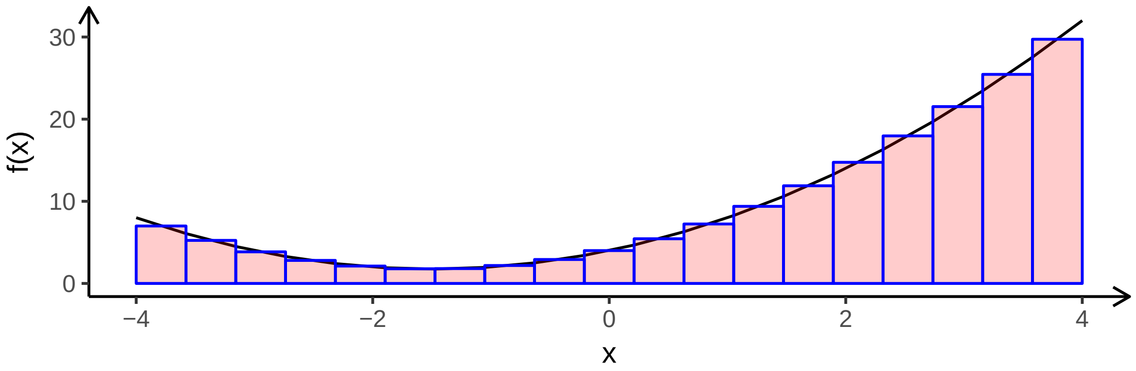

Figure 1: Illustration of the Midpoint or Rectangular Quadrature

The midpoint rule is exact for linear functions. Please note, that $E(f)$ might also be used up front to determine the needed number of subintervals n in order to not exceed a certain level of accuracy.

3.2.2 Simpson-Rule

The Simposn-Rule is similiar to the Midpoint Quadrature, but instead of rectangles quadratic functions are used in order to calculate the area between the x-axis and our limited function $f : [a; b] \rightarrow \mathbb{R}$ 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 $g_i(x) : [x_i; x_{i+1}] \rightarrow \mathbb{R}$. For our approximation it then holds that $\int_a^b f(x)dx \approx \frac{\Delta x}{6} \cdot \left(f(x_0) + 2 \cdot \sum_{k=1}^{n-1} f(x_k) + f(x_n) + 4 \cdot \sum_{k=0}^{n-1} f(x^{(k)})\right)$.

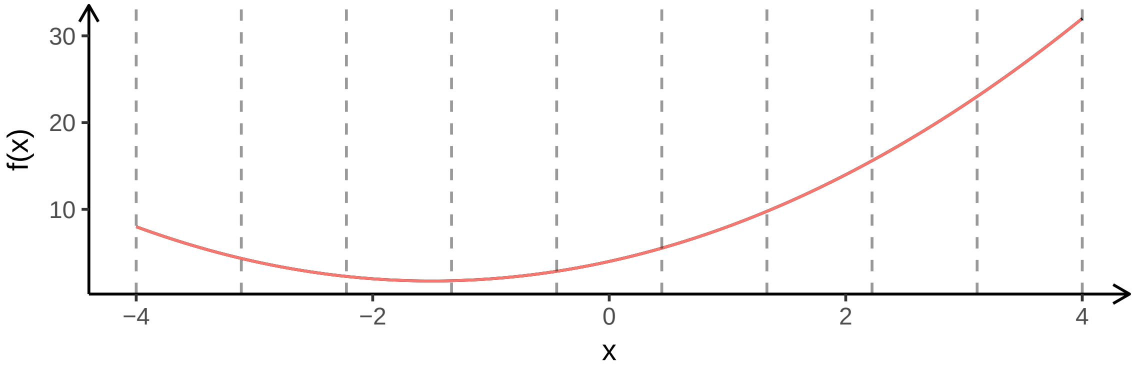

Figure 2: Illustration of the Simpson Rule

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 $E(f)$. Again, according to (Davis J. Philip

2007, 57), 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:

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

The Simpson rule is exact for all polynomials of degree three or less. As in the case of the midpoint quadrature, one might use $E(f)$ to determine the needed number of subintervals n in order to not exceed a certain threshold of error before applying the algorithm.

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 an 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. The procedure works for a limited function $f : [a;b] \to \mathbb{R}$ as stated by the pseudocode below:

```
AdaptiveRule($f$, $[a;b]$, $\varepsilon$)
    Set $c = \frac{b-a}{2}$ and split $[a;b]$ into $[a;c]$ and $[c;b]$
    Calculate $I_1 \approx \int_a^c f(x)dx$ and $I_2 \approx \int_c^b f(x)dx$
    If $\varepsilon_{I_1} + \varepsilon_{I_2} \leq \varepsilon$ then
        Return $I_1 + I_2$
    Else
        Return AdaptiveRule(($f$, $[a;c]$, $\frac{\varepsilon}{2}$) + AdaptiveRule(($f$, $[c;b]$, $\frac{\varepsilon}{2}$)")
```

It may be noted, that the crucial parts of the adaptive algorithm are the quadrature method (e.g. Simpson or Midpoint) and the corresponding error estimator. Visualized the procedure looks as follows:

It may be noted, that the adaptive algorithm works just as fine as the algorithms elaborated on above in case of "well behaved" functions, but outperforms when we want to numerically integrate "badly behaved" functions. One example of functions that are not well behaved and require for an adaptive quadrature method are functions which rise/fall suddenly in a strong manner.

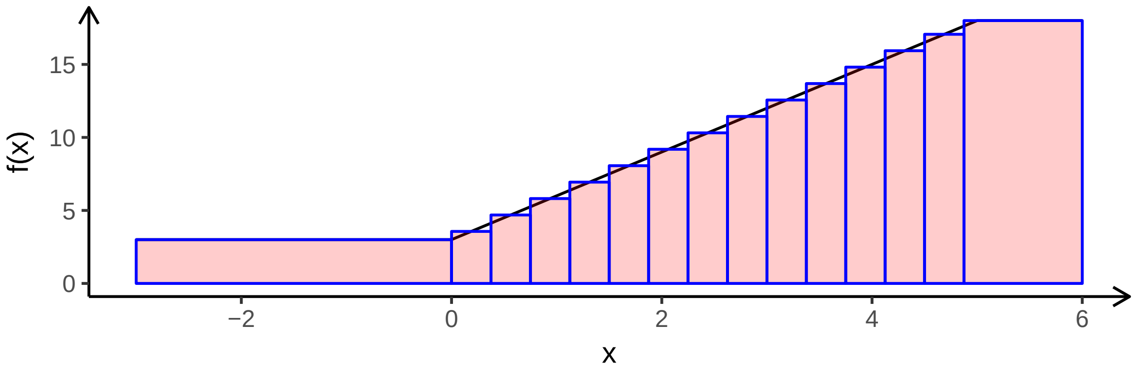

Figure 3: Illustration of the adaptive Midpoint Quadrature

3.3 Multi-Dimensional Procedures

In lots of applications multidomensional Integrals need to be solved numerically. This means that we now want to calculate the integral $\int_{\Omega} f(\mathbf{x}) d\mathbf{x} = \int_{a_1}^{b_1} ... \int_{a_m}^{b_m} f(x_1, x_2, ..., x_m) dx_1 dx_2 ... dx_m$. 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** (see appendix for an example illustration). Therefore I want to concentrate on Monte Carlo integration methods going on. It may be noted, that both methods presented here result in unbiased estimator of our integral of interest. Please note that the goodness of your approximation depends on your pseudorandom number generator when applying Monte Carlo integration methods. Furthermore, for illustration we use one-dimensional integrals for simplicity, but according to (Gentle 2003, 233) Monte Carlo integration methods should ordinarily only be used for multi-dimensional integrals.

3.3.1 Crude

The crude monte carlo integration method is a fairly simple method to calculate integrals numerically. One needs to generate n m-dimensional uniformally distributed points $\mathbf{x_i} = (x_1, ..., x_m)\ \forall i \in \{1, ..., n\}$. These points are uniformly distributed on $[a_1; b_1] \times ... \times [a_m; b_m]$. In the next step, we calculate $f(\mathbf{x_i})\ \forall i \in \{1, ..., n\}$ and the corresponding mean $s = \frac{1}{n} \sum_{i=1}^{n} f(\mathbf{x_i})$. Therefore it holds that $\int_a^b f(x) dx \approx s \cdot (\prod_{l=1}^{m} (b_l - a_l))$

As stated before, Monte Carlo is a probabilistic approach and therefore the questions of bounds for the error of approximation does not arise. Instead of error bounds, we use the variance of the random estimator to indicate the extent of the uncertainty in our approximation. Let $\hat{\theta}_{CMC} = s \cdot (\prod_{l=1}^{m} (b_l - a_l))$ be our estimator for the integral and $V = \prod_{l=1}^{m} (b_l - a_l)$ be the volume of our base area. Then the following holds according to (Gentle 2003, 231):

$$\hat{\mathbb{V}}[\hat{\theta}_{CMC}] = \frac{V^2}{n} \hat{\mathbb{V}}[f] \qquad with \qquad \hat{\mathbb{V}}[f] = \frac{1}{n-1} \sum_{i=1}^{n} (f(x_i) - m)^2$$

According to (Davis J. Philip 2007, 388 - 417) crude monte carlo exists as a last resort,

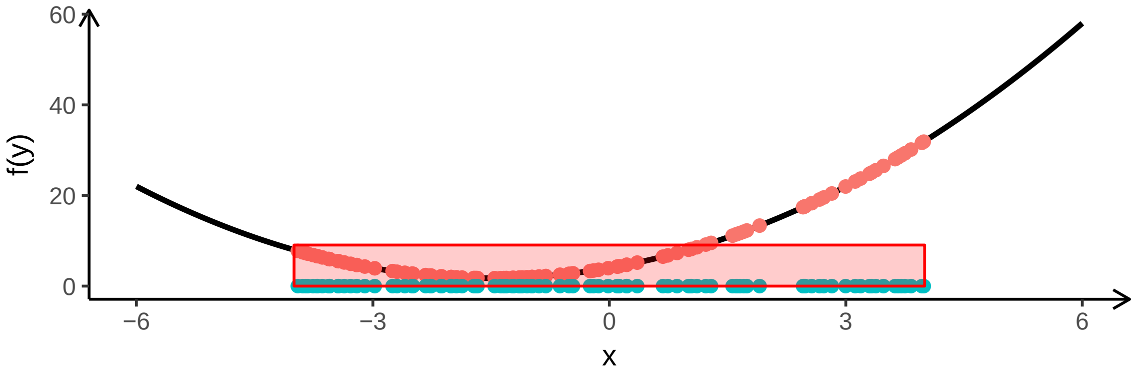

Figure 4: Illustration of the Crude Monte Carlo integration method

especially for integrals over nonstandard domains or integrands of low order continutity. Another main advantage is, that the rate of convergence is independent of the smoothness of the integrand.

3.3.2 Hit or Miss

The hit or miss monte carlo integration method works slidly differently compared to the crude monte carlo integration. Here we create n $m+1$-dimensional uniformly distributed points $(x_{i,1},...,x_{i,m},y_i)$ for $i \in \{1,..,n\}$. 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 the i^{th} coordinate is uniformaly distributed on $[a_i;b_i]$ $\forall i \in \{1,...,m\}$. The y_i coordinate corresponds to the functions range and is 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/volume V surrounding the graph needs to be calculated. Therefore it holds that $\int_a^b f(x)dx \approx V \cdot p$ with $V = max(f(\mathbf{x})) \cdot \prod_{l=1}^{m}(b_l - a_l)$.

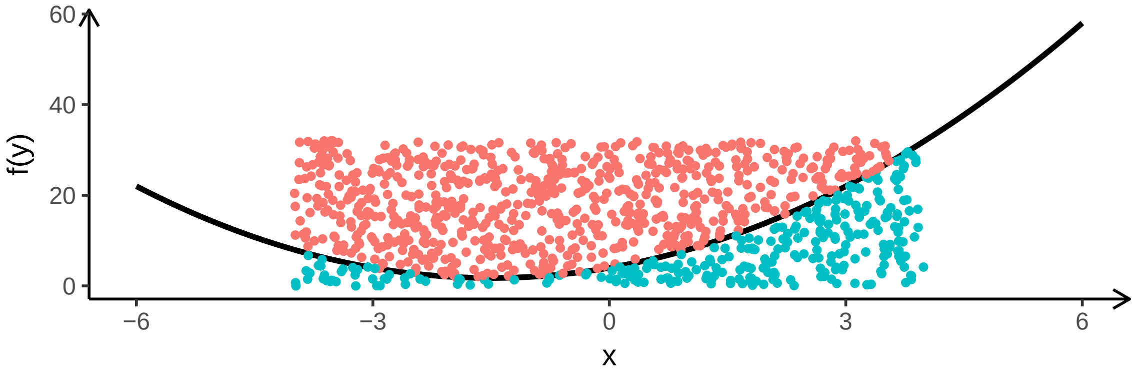

Figure 5: Illustration of the Hit or Miss Monte Carlo integration method

As for the crude monte carlo integration we are also interested in the variance of our estimator $\hat{\theta}_{HM} = V \cdot p$ for which holds:

$$\hat{\mathbb{V}}[\hat{\theta}_{HM}] = \frac{V^2 \cdot p(1-p)}{n}$$

It may be noted, that according to (Hammersley 1964, 54) and (Gentle 2003, 232) the crude monte carlo method is the superior method, since both estimators are unbiased, but the variance of $\hat{\theta}_{CMC}$ is always smaller for fixed n.

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 distributions density function since it cannot be expressed using only elementary functions. In order to do so I will use the scientific programming language R. 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 as a method of choice. Since I need to solve an improper integral a change of variables is used and furthermore I will apply $15 \cdot i$ bins for each sampling point x_i with $i = 1, .., 12$.

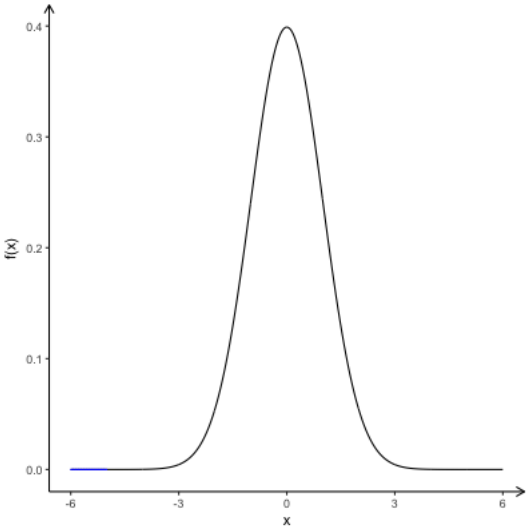

Figure 6: Visualization of the steps of integration

To calculate $F(x)$ we now need to solve the following equation for our $N = 12$ sampling points:

$$y_i = F(x_i) \approx \int_{-\infty}^{x_i} f(x)dx \;\; \forall i \in \{1, .., 12\}$$

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:

$$\hat{F}(x) = \frac{1}{1 + \exp^{-\hat{b}\cdot(x-\hat{c})}} = \frac{1}{1 + \exp^{-1.706\cdot x}}$$

After approximating our function we now need to test the goodness of fit. We will do this by applying a collection of normality tests. I will use the Shapiro-Wilks, Anderson-Darling, Lilliefors and Kolmogorov-Smirnov as tests of choice, even though (Razali 2011) showed, that Shapiro-Wilks is the most powerful of the tests used. In order to test whether there is evidence that our approximated distribution is not normal, we need to sample datapoints. For this we create 500 uniformly on $[0; 1]$ distributed random samples. Then we use the inverse of our fitted function to have sample datapoints corresponding to the approximated CDF $\hat{F}$. The corresponding results/p-values can be found below.

Table 2: Results/p-values of the normality tests

Shapiro-Wilks:	Anderson-Darling:	Lilliefors:	Kolmogorov-Smirnov:
0.347	0.2165	0.2015	0.4047

According to the test's p-values we cannot reject the null hypothesis even ones, meaning we could not find enough evidence to conclude, that the datapoints are not normally distributed (corresponding QQ-Plot can also be found in appendix). 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.

The points displayed on the approximated curve $\hat{F}$ are the 500 randomly sampled datapoints.

5. Conclusion

The 5 procedures elaborated on in this document are a good basis in numerical integration. In our application one could for example easily see, that even though just basic methods were applied we can already easily retrieve "good" results. It may be noted, that "good" in this sense means, that we are not able to reject the null hypothesis for some of the most well known normaility tests. Depending on the application at hand this measure of goodness might not be appropriate and the user therfore might have to use more advanced methods in order to achieve the desired level of goodness respectively accuracy. Due to the fact that the topic is very important in applications, there were many more procedures developed by the scientific community and each of these comes with its own advantages and disadvantages. For

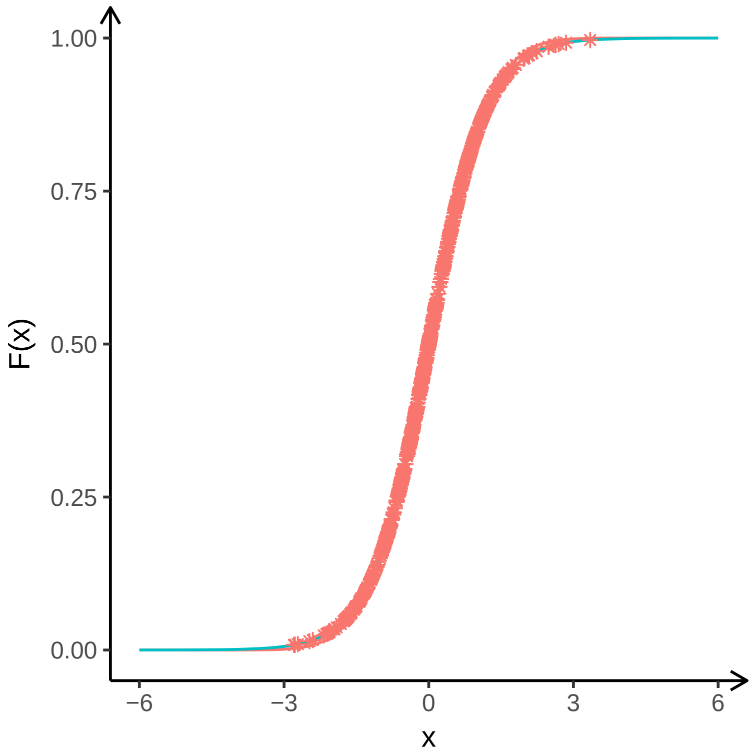

Figure 7: Visualization of the fitted (red) and R's Normal Distributions CDF (blue)

further readings on the topic one might for example proceed with the textbooks elaborated on in the literature review or bibliography.

I Appendix

Bibliography

Davis J. Philip, Rabinowitz Philip. 2007. *Methods of Numerical Integration*. Dover Publications.

Gentle, James E. 2003. *Random Number Generation and Monte Carlo Methods*. Springer-Verlag New York.

Hammersley, Handscomb, John Michael. 1964. *Monte Carlo Methods*. Springer Netherlands.

Kythe, Schaferkotter, Prem K. 2005. *Handbook of Computational Methods for Integration*. Chapman & Hall/CRC Press.

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

Press, et al., William H. 2007. *Numerical Recipes - the Art of Scientific Computing*. New

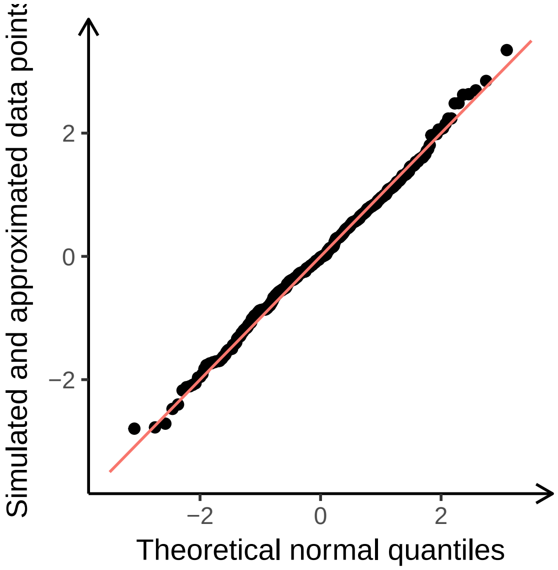

Figure 8: QQPlot of R's Normal Distributions CDF and the simulated points distributed according to the approximated CDF

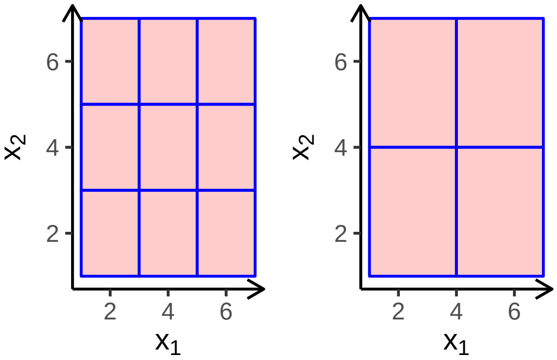

Figure 9: Example illustration of the curse of dimensionality

York: Cambridge University Press.

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.

Rizzo, Maria L. 2007. *Statistical Computing with R*. Chapman; Hall/CRC.