Technische Universität München
Fakultät für Physik

Bachelor's Thesis in Physics

Novel Algorithms for Data Analysis

Evidence Calculation with Markov Chain Monte Carlo-Methods

Stephan Kaltenstadler

Garching, 12. August 2013

Advisor: Prof. Dr. Allen Caldwell
Second Advisor: .....

Contents

List of Figures

Introduction

Modern empirical science and especially experimental physics are confronted with the evaluation of highly complex data received through their measurements. In many cases it is necessary to rely on computational methods for this task.
The Bayesian Analysis Toolkit,BAT, is a C++ Toolkit[1] which enables the user to perform data analysis based on statistical Monte Carlo methods. This analysis is realised using the Bayes Theorem

$$P(\theta|D,M) = \frac{P(D|\theta,M)P_0(\theta|M)}{P(D|M)} \tag{1}$$

that determines the probability of a parameter for given data, D, and model, M. Although BAT is capable of this kind of operation, it lacks a robust method to calculate the evidence,

$$Z = P(D|M) = \int P(D|\theta,M)P_0(\theta|M)d\theta \tag{2}$$

which is the probability of the measured data for our model.
We observe a function in a multi-dimensional parameter space, which makes it difficult to evaluate Z with standard numerical integration routines. The problem is usually that in a multidimensional space the regions where our likelihood function $P(D|\theta,M)$ is large occupies a vanishingly small volume in the space covered by the prior $P_0(\theta)$.
There are several approaches to solve this integral (2), such as nested sampling[2], to directly evaluate the integral, or estimate the probability density of the posterior at the mode and using $Z = \frac{P(D|\theta^*)P_0(\theta^*)}{P(\theta^*|D)}$[3].
In this theses we will try a different approach where we use a MCMC-method to produce sample according to our likelihood, $P(D|\theta,M)$, which gives us the regions where the likelihood is large and then calculate our integral for these limited volumes, instead of the whole parameter space, with a sample mean approach:

$$Z = \int P(D|\theta,M)P_0(\theta|M)d\theta = \frac{1}{N}\sum_{i=1}^{N} P(D|\theta,M)|_{P_0(\theta|M)} \tag{3}$$

[1] [Caldwell,Kollar,Kröninger]
[2] [Skilling]
[3] [Chib,Jeliazkov]

Chapter 1

Formulation

We assume we have successfully run a MCMC to extract sample according to $P(\theta|D, M)$. We can estimate the global mode from this MCMC, so we know where the integrand of our evidence (2) has it's maximum. Now we rewrite Bayes' equation

$$Z \cdot P(\theta|D, M) = P(D|\theta, M) \cdot P_0(\theta, M) \tag{1.1}$$

and integrate both sides over a limited volume centered on the mode:

$$\int_{\Delta\theta} Z \cdot P(\theta|D, M) d\theta = \int_{\Delta\theta} P(D|\theta, M) \cdot P_0(\theta, M) d\theta \tag{1.2}$$

and get

$$Z = \frac{\int_{\Delta\theta} P(D|\theta, M) \cdot P_0(\theta, M) d\theta}{\int_{\Delta\theta} P(\theta|D, M) d\theta}. \tag{1.3}$$

The denominator, which we will call estimator from now on, is given directly by the existing MCMC output:

$$\int_{\Delta\theta} P(\theta|D, M) d\theta \approx \frac{N_{\Delta\theta}}{N_{Total}} \tag{1.4}$$

where $N_{\Delta\theta}$ is the number of posterior samples falling within region $\Delta\theta$ and N_{Total} is the total number of samples of our Markov chain.
Now we can evaluate the numerator of equation (1.4) with a sample mean technique like in (3) but now over a reduced region $\Delta\theta$. Therefore we define a hypercube centered around the mode in the parameter space which gives us $\Delta\theta$. The numerator in (1.4) is then

$$\int_{\Delta\theta} P(D|\theta, M) \cdot P_0(\theta|M) d\theta \approx \frac{1}{K} \eta \sum_{i=1}^{K} P(D|\theta, M)|_{P_\Delta(\theta)}. \tag{1.5}$$

with $P_\Delta(\theta)$ being the prior probability in the region of our hypercube. The factor $\eta = \frac{\int_{\Delta\theta} P_0(\theta) d\theta}{\int P_0(\theta) d\theta}$ is necessary because we need to renormalize our prior for the reduced

parameter space. For a flat prior like we use here $\eta = \frac{V_{\Delta\theta}}{V_{total}}$
In this thesis we will try this approach on four different examples:

```
1 Gaussian in one  dimension
2 Gaussian in higher dimensions (example will be 10)
3 Gaussian Shells
4 Gaussian Shells in higher dimensions (10)
```

For the gaussians we will create some "fake-data"and calculate the evidence for it, while the gaussian shells are just a example with a degenerated mode to test the algorithm for this case.

Chapter 2

Theoretical Principles

2.1 Monte Carlo Methods

Monte Carlo methods are stochastic methods that rely on random sampling to simulate solutions in a numerical way which would be difficult or even impossible to calculate analytically. The basic concept is to simulate an experiment with random numbers instead of data. Monte Carlo methods are especially effective in solving very complex high dimensional integrals wtih the sample mean integration technique which is also called Monte Carlo Integration.
The sample mean integration is based on the mean value theorem for integration. This theorem shows, that the integration of a function f over a region A is equivalent to the mean value of the function on A multiplied with the volume of A.

$$\int_A f(x) = \bar{f}(x)A \tag{2.1}$$

The mean value $\bar{f}(x)$ can be identified by a summation over the whole area A.

$$\bar{f}(x) = \frac{\sum_{m=1}^{N} f(x)}{N} \tag{2.2}$$

with x being randomly distributed over A. Using this we are able to replace the integration through a summation which is much easier to perform computationally.

2.2 Markov Chains

Markov Chains are a technique to produce a set of random variables or states, which fullfil certain properties: They are memoryless and they can be defined by an transition kernel.[1] For the the discrete case, which is used in this thesis, these two

[1] l[Robert,Casella,2004, p.208]

conditions can be formulated quite simply: The transition kernel simplifies to an matrix defined by:[2]

$$P_{xy} = P(X_n = y | X_{n-1} = x). \tag{2.3}$$

Memoryless or independent means that the transition probability is dependant only on the current position:[3]

$$P(X_k + 1 \in A | x_0, x_1, x_2, ..., x_k) = P(X_k + 1 \in A | x_k) \tag{2.4}$$

We will just use the regular case of Markov chains, which means that for any two states *i,j* there is a non-zero probability for a transition from i to j . Simply put, this means that for an infinite number of transitions, the Markov chain should reach every state.
One of the most common applications of Markov chains is the so called Metropolis algorithm.

2.3 The Metropolis Algorithm

The Metropolis algorithm or it's general form, the Metropolis-Hastings algorithm is a basic tool in Monte Carlo methods.The goal of this algorithm is to produce random numbers which follow an special distribution or function, that we call *f*. To achieve this we use the following accept-reject method. The current state is x_i, y is a different state produced by a symmetric random process.

Algorithm 1 Metropolis algorithm

Produce random − state y
define $R = \frac{f(y)}{f(x_i)}$
if $R \geq 1$ **then**
  $x_{i+1} = y$
else
  create random variable U between 0 *and* 1
  if $R \leq U$ **then**
    $x_{i+1} = y$
  else
    $x_{i+1} = x_i$
  end if
end if

[2][Robert,Casella,2004, p.208]
[3][Robert,Casella,2004, p.209]

By repeating this method we create a set of random numbers which is distributed similar to f.

Chapter 3

Results and discussion

3.1 Procedure

The approach for the given tasks is quite similar, we start with the production of random numbers according to the given function and dimension, by using the Metropolis algorithm described above.

Algorithm 2 Data generation

define f
create random-state y
define $R = \frac{f(y)}{f(x_i)}$
for $i = 1 \rightarrow n$ **do** ▷ pre-run to achieve convergence of the markov chain, chose n accordingly
    $x_{i+1} = Metropolis(x_i, f)$
end for
for $i = 1 \rightarrow Number\ of\ Data$ **do**
    $x_{i+1} = Metropolis(x_i, f)$
    save x_{i+1}
end for

The next step is to calculate the evidence for the produced data analytically, which is possible because we work on known functions in this paper. This is necessary because we analyze the quality of the our algorithm by comparing it to the true value.
Then follows a first numerical calculation of the evidence, we determine the evidence with a sample mean integration over the full parameter size. We evaluate the accuracy of this integration dependant on the size range. The result of this integration should be accurate, but this technique is very ineffective, especially if we use it in higher dimensions.

The last step is to use our new method: For this purpose we use a Metropolis algorithm similar to the one for the data generation, but this time we have data and therefore a different likelihood-function

$$f^* = \prod_{i=1}^{N} f(\theta, D_i) \tag{3.1}$$

which is the product of our single likelihood-functions for every single piece of data D_i with the number of data samples N.
To be able to run this reduced approach we need to initialize a markov chain to find our region and our mode, before we can apply a sample mean calculation over the reduced area.

Algorithm 3 Markov chain to find mode, region and estimator

load data
create random-parameter-state y
define $f^*(\theta) = \prod_{i=1}^{Numberofdata} f(\theta|x_i)$ ▷ x_i = datapoint

define *mode*
for $i = 1 \rightarrow iterations$ **do** ▷ quality of our Markov chain increases with number of iterations
  $\theta_{i+1} = Metropolis(\theta_i, f^*)$
  save θ_{i+1}
  if $f^*(mode) \leq f^*(\theta_i)$ **then**
    $mode = \theta_i$
  end if
end for

define $counter = 0$
define *Region* ▷ hypercube centered on the mode
for $i = 1 \rightarrow n$ **do**
  if $\theta_i \in region$ **then**
    $counter+ = 1$
  end if
end for
define $estimator = \frac{counter}{iterations}$

Algorithm 4 Sample mean over reduced area

define $Sum = 0$
for $i = 0 \rightarrow K$ **do** ▷ K=number of samples
create Random-value y $\in$ region
$sum+ = f^*(y)$
end for
$Sample\ mean = \frac{sum}{K}$

According to our instruction we can now eventually calculate our evidence:

$$Z = \frac{sample\ mean \cdot volume_{region}}{volume_{Parameter\ space}} \tag{3.2}$$

The two entities we are most interested here are the ratio between our reduced sample mean value and the reference value and the behaviour of the estimator $\frac{N_{\Delta\theta}}{N_{Total}}$.

3.2 1-dimensional Gaussian

We start with applying our algorithm to a simple problem, the single-dimensional Gaussian-function. This function has two parameters, the expectation value μ and the variance σ which makes it a two dimensional problem. Here we will discuss this two dimensional problem and also a reduction on a single dimension where we will take σ as fixed.
Calculations over a more dimensional parameter spaces need greater computational efforts and optimization of our markov chains. In the formulation we assumed that the markov-chain has run successfully. So we can focus on the post-processing and therefore it is valid to use simple parameter spaces, like the reduction on the expecation value. Nethertheless we will show for the single dimensional gaussian that the algorithm will in principle also work for more paramters.

3.2.1 Data generation and determination of the reference value

We set our parameters to μ=0 and $\sigma = 1$ and produce the dataset with a Metropolis algorithm according to this. In the data analysis step, our maximal paramter space is $[-100; 100]$ for μ and $[0.1; 10]$ for σ.
To determine a reference evidence for our algorithm, we will use a analytical approximation and a normal sample mean integration over the whole parameter space with a great amount of samples.

A analytical approximation for the evidence is

$$Z = \frac{1}{\Delta_\mu \Delta_\sigma (2\pi)^{\frac{d-1}{2}} d^{\frac{1}{2}}} \frac{(\frac{d}{2}-2)!}{2(\frac{dS_{x^2}-S_x^2}{2d})^{\frac{d}{2}-1}} \tag{3.3}$$

with $S_{x^2} = \sum_{i=1}^{d} x_i^2$ and $S_x = \sum_{i=1}^{d} x_i$, d the number of dimensions, which is the number of datasets.
The evidence for 1 paramter is the following:

$$Z = \frac{1}{\Delta_\mu (2\pi)^{\frac{d-1}{2}} \sigma^{d-1} \sqrt{d}} \exp - \frac{dS_{x^2} - S_x^2}{2d} \tag{3.4}$$

The derivations of these analytical formulas are shown in appendix A and B.

For 2-dimensional paramter space the analytical approximation delivers the value $1.08606e-25$ while a sample mean calculation with $1e+8$ iterations delivers $1.09038e-25$.
The results for analytical and sample mean are comparable,the difference is about 1%. We will take the sample mean result as our "true"result which we compare to the result of our algorithm.

For the case with both parameters we want to describe our region with a single variable delta, therefore we must transform our intervall of the variance between [0.1;10] into the intervall [-100;100] of the expectation value to make their parameter space comparable. We define the variable σ^*

$$\sigma^* = 100 \log_{10} \sigma \tag{3.5}$$

which is now distributed in the intervall [-100;100].

3.2.2 1-dimensional gaussian without variance

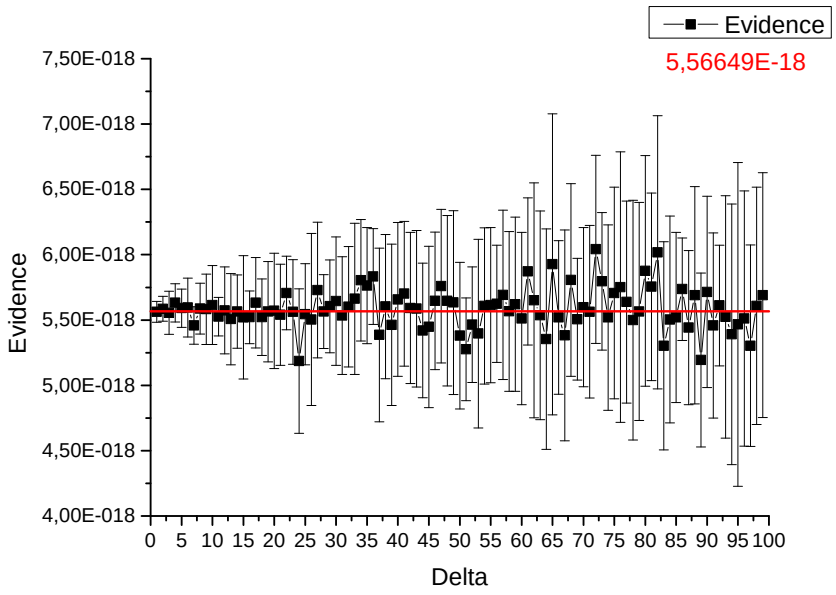

Figure 3.1: Evidence for 1-dimensional gaussian without variance for a delta-region between 0 and 100. The dataset consists of 20 samples, the algorithm was run 15 times to determine mean value and standard-deviation

Figure 3.1 shows the calculated evidence of the 1-dimensional gaussian reduced on the parameter μ. For a small area delta, we get valid results and a very small deviation, even for the very small number of sample mean iterations of 1000. For bigger deltas we lose our advantage of a small region and descent into a normal sample mean calculation with bad precision because we use just 1000 samples. Figure 3.2 shows the bevaiour of the evidence for very small delta, we see that here our precision gets much better than for bigger delta-areas. Only for very small delta<0.1 our accuracy gets worse again, so if we look at the estimator in figure 3.3 and it's relation with our precision in figure 3.4, we see that we do not necesserily need a very big ratio of samples in our area, altough precision increases slightly.

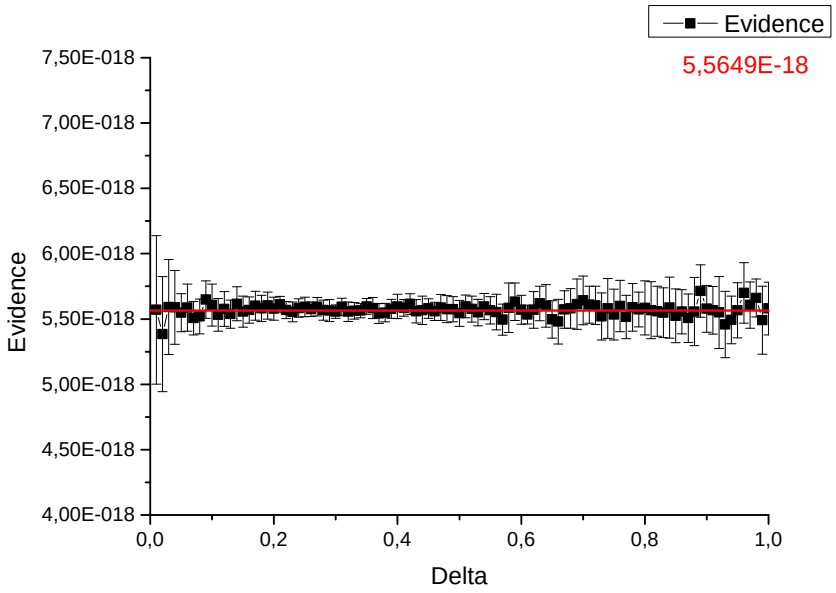

Figure 3.2: Evidence for 1-dimensional gaussian without variance for a delta-region between 0 and 1 with a sample mean of 1000 samples. The dataset consists of 20 samples, the algorithm was run 15 times to determine mean value and standard deviaton

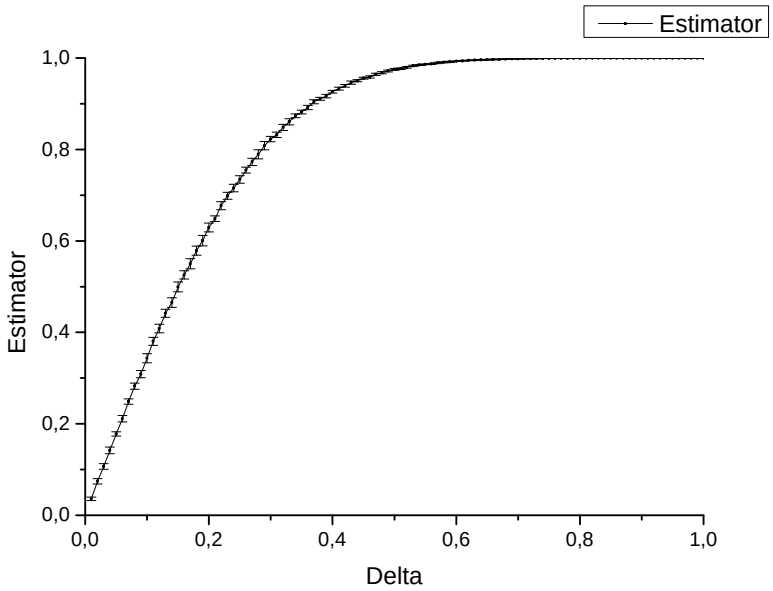

Figure 3.3: Estimator for 1-dimensional gaussian without variance for a delta-region between 0 and 1 with a MCMC of 100000 samples. The dataset consists of 20 samples, the algorithm was run 15 times to determine mean value and standard-deviation

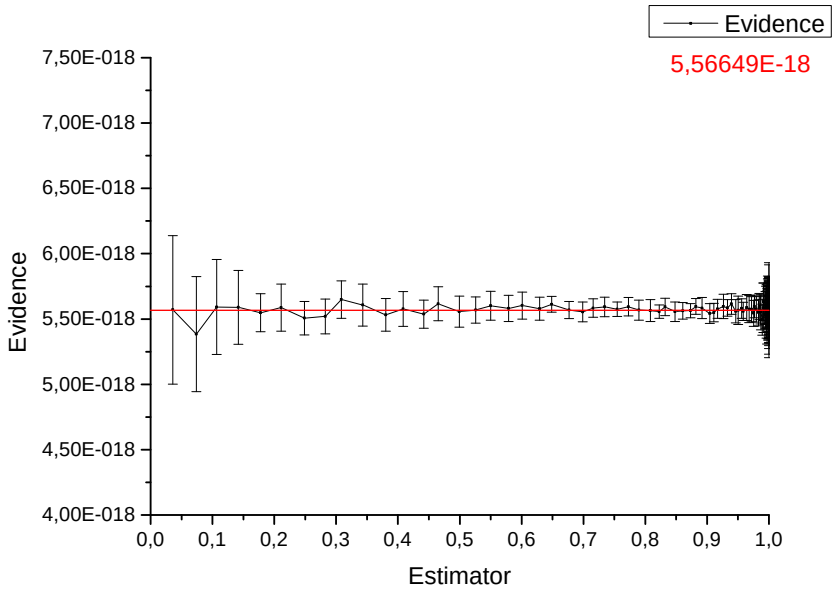

Figure 3.4: Estimator for 1-dimensional gaussian without variance for a delta-region between 0 and 1 with a sample mean of 1000 samples. The dataset consists of 20 samples, the algorithm was run 15 times to determine mean value and standard-deviation

3.2.3 1-dimensional gaussian with variance

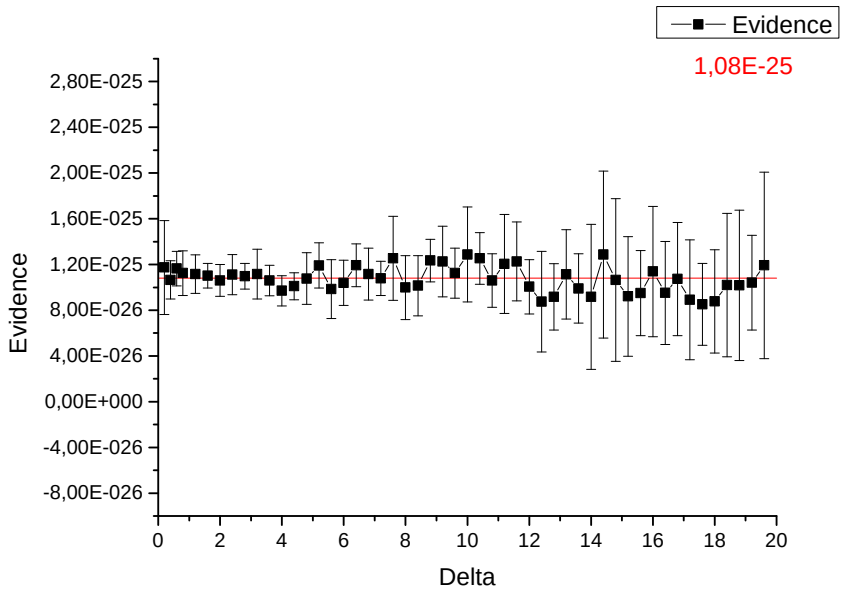

Figure 3.5: Evidence for 1-dimensional gaussian with variance for a delta-region between 1 and 20 with a sample mean of 1000 samples. The dataset herefore consisted of 50 samples, the algorithm was run 10 times to determine mean value and standard-deviation.

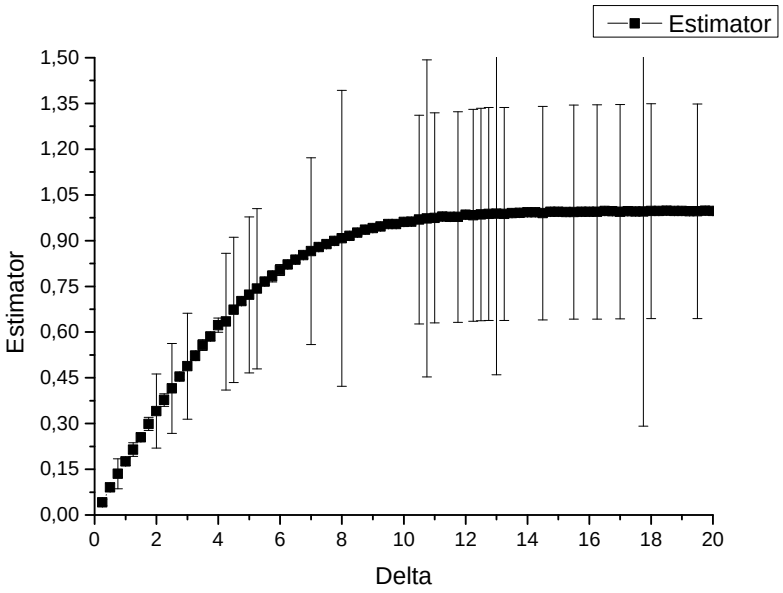

Figure 3.6: Estimator for 1-dimensional gaussian with variance for a delta-region between 1 and 20 from a MCMC with 100000 samples. The dataset herefore consisted of 50 samples, the algorithm was run 10 times to determine mean value and standard-deviation.

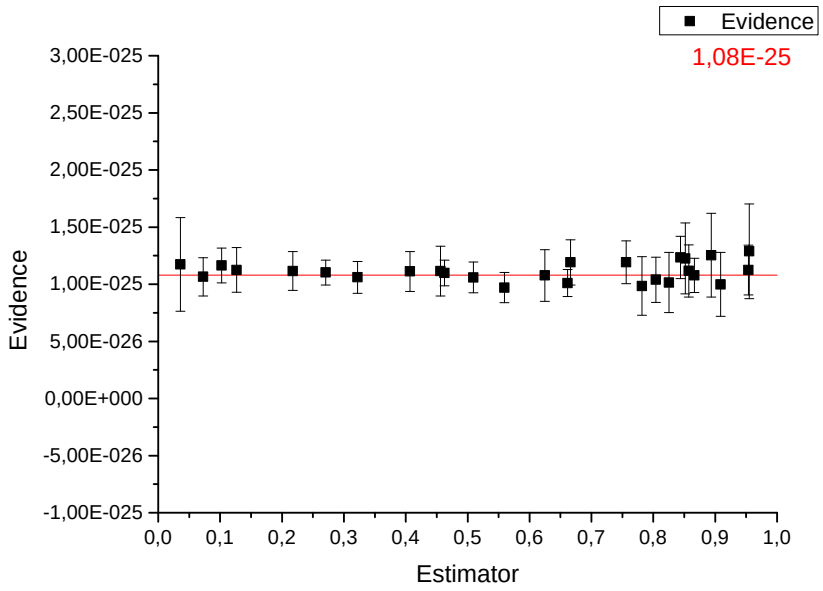

Figure 3.7: Evidence for 1-dimensional gaussian with variance for a delta-region between 1 and 20 with a sample mean of 1000 samples. The dataset herefore consisted of 50 samples, the algorithm was run 10 times to determine mean value and standard-deviation.

In Figure 3.5 we see the 1-dimensional gaussian result where both the mean and the variance are free parameters on delta values between 0 and 20. This region was chosen because it is the region where the estimator variates, as shown in figure 3.6. We see again that for small delta, especially delta ≤ 5 the algorithm delivers a valid result with a small deviation. For big delta our method tends to a normal sample mean in which case 1000 samples for a region this big are not enough to achieve a good result. Figure 3.7 shows that there is almost no influence of the size of the estimator on the precision of the evidence calculation.

3.3 Gaussian in 10 dimensions

After we have shown the functionallity for our algorithm in principle, we now apply it on a first complex problem: the gaussian function in 10 dimensions:

$$f(x|\mu,\sigma) = \frac{1}{(2\pi)^{\frac{N}{2}}|\Sigma|^{\frac{a}{2}}} e^{-\frac{1}{2}(x_i-\mu)^T\Sigma^{-1}(x_i-\mu)} \tag{3.6}$$

We start again with creating our data according to our parameters, which are chosen similar to the single dimensional problems: $\mu = (0,0,0,0,0,0,0,0,0,0)$, $\sigma = (1,1,1,1,1,1,1,1,1,1)$, the parameter space for μ_i is [-100;100] in every dimension. The analytical expression according to appendix B is

$$Z = \frac{1}{(\Delta_\mu^2 10)^{\frac{10}{2}}} \frac{1}{(2\pi)^{\frac{90}{2}}} e^{-\frac{10}{2}(\sum_{j=1}^{10} Var[x_j])} \tag{3.7}$$

which gives us 1.05535e-68 as evidence for our dataset. We cannot give a result from normal sample mean calculation because even with 1e+8 samples the sample mean method gives results which differ by several orders of magnitude.
As we see in figure 3.8, the algorithm works even in this case as long we find the right delta region for the algorithm which corresponds roughly with the region where the estimator variates, like it is visible in figure 3.9. In this region, there is almost no correlation between the size of the estimator and the qualitiy of our result. On contrary to the previous low-dimensional cases the algorithm does not work if we leave this region.

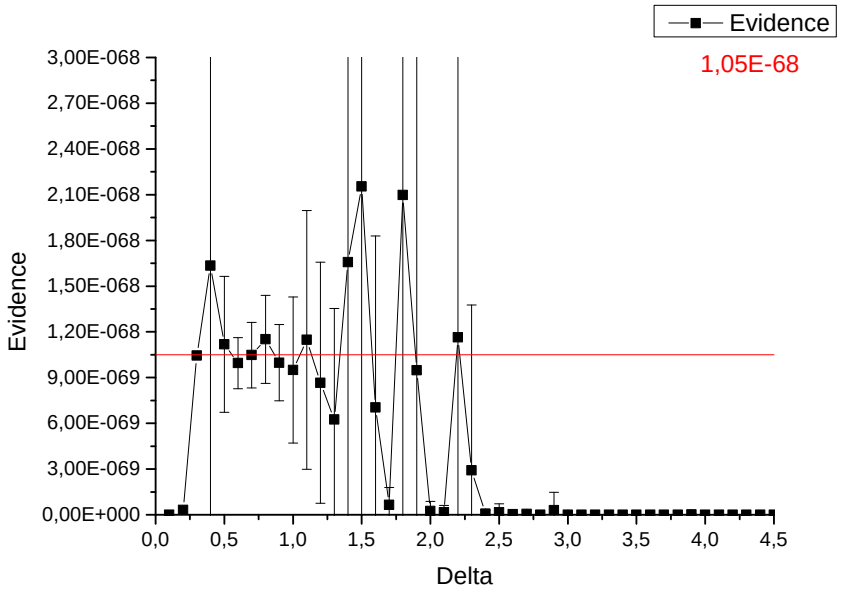

Figure 3.8: Evidence for 10-dimensional gaussian shell for a delta-region between 0 and 4.5 with a sample mean of 1000 samples. The algorithm was run 10 times to determine mean value and standard-deviation.

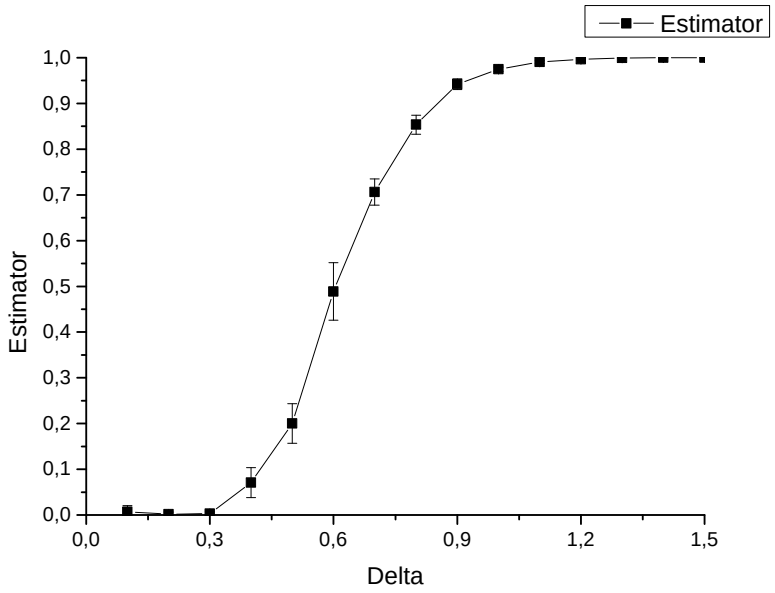

Figure 3.9: Evidence for 10-dimensional gaussian shell for a delta-region between 0 and 4.5. The algorithm was run 10 times to determine mean value and standard-deviation.

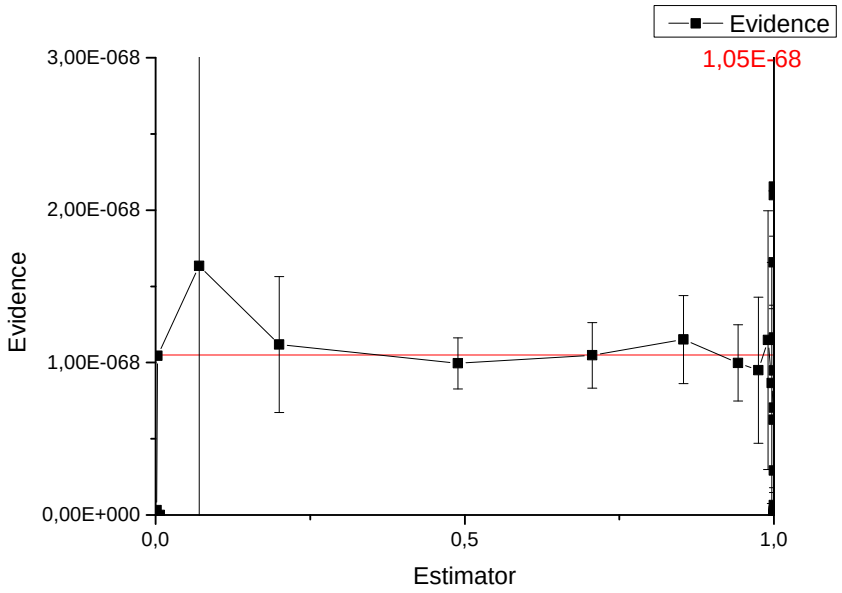

Figure 3.10: Relation between evidence and estimator for 10-dimensional gaussian shell for a delta-region between 0 and 4.5. The algorithm was run 10 times to determine mean value and standard-deviation.

3.4 Gaussianshell in 2 dimensions

Now we look at another likelihood function which is called Gaussian-shell:

$$f(\theta|x,c,r) = \frac{1}{\sqrt{2\pi w^2}} exp - \frac{(|\theta - c| - r)^2}{2w^2} \tag{3.8}$$

This is a circle which is decreases with a gaussian around the mode. The parameters we use here are radius $r = 5$, width $w = 2$ and the center point $c = (0, 0)$.
For this function we do not create any false data, we just use it to test the behaviour of our algorithm for the case of a degenerated mode, which in this case is a circle instead of the points we used to examine for the gaussian-functions. The analytical evidence for this problem is given with

$$Z = \frac{\sqrt{2}\pi^{(\frac{d-1}{2})}}{\Gamma(\frac{d}{2})(2\rho_{max})^d w} \int_0^{\rho_{max}} d\rho \rho^{d-1} e^{-\frac{(\rho-r)^2}{2w^2}} 1 \tag{3.9}$$

[1] [Beaujean,Caldwell, 2013, p.8]

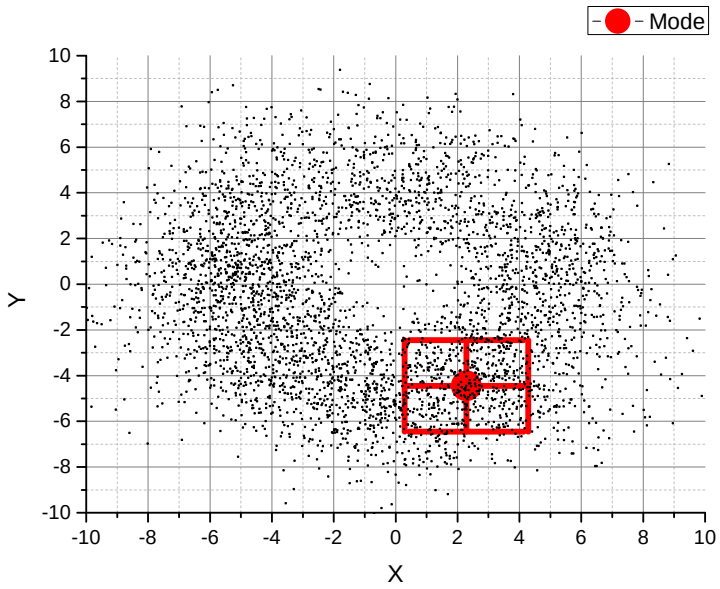

Figure 3.11: Example for our delta area for gaussian shell in 2 dimensions. The area is the red square with an edge length of 2 delta around the mode. The black points are results of the MCMC from gaussian shell with radius 5, delta was 2.

d as the number of dimensions.
For our true value we take again sample mean and analytical calculations: The sample mean with $1e+8$ iterations gives the result $7.856e-4$ for the evidence, the analytical integral gives $7.86e-4$.
The evidence calculation for delta-values between 0 and 25 with the reduced sample mean is shown in figure 3.12. The precision is optimal for delta-areas between 1 and 10, for big deltas it gets worse again as in the previous examples. If we look at our estimator in figure 3.13 and the relation between estimator and evidence in 3.14, we see that there is almost no influence of the estimator on the precision of our evidence as long it is not vanishingly small as for $delta \leq 1$.

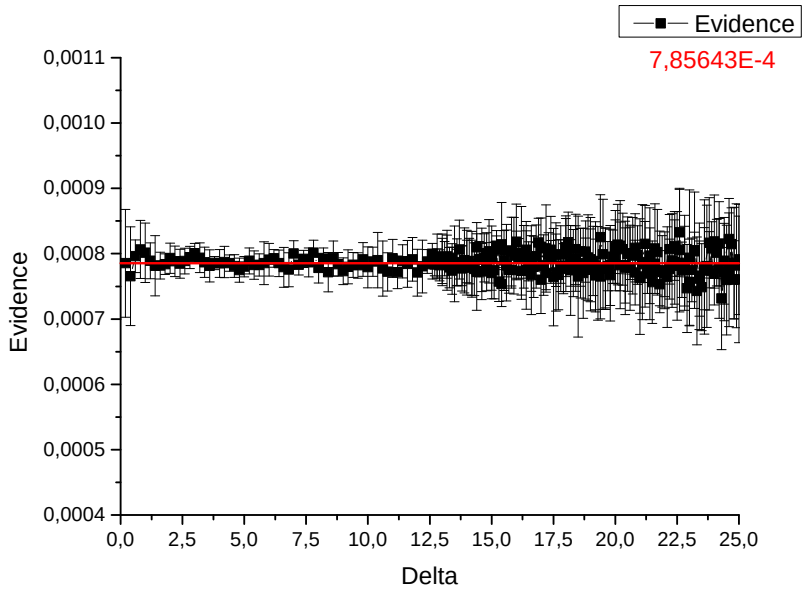

Figure 3.12: Evidence for 2-dimensional gaussian shell for a delta-region between 0 and 25 with a sample mean of 1000 samples. The algorithm was run 10 times to determine mean value and standard-deviation.

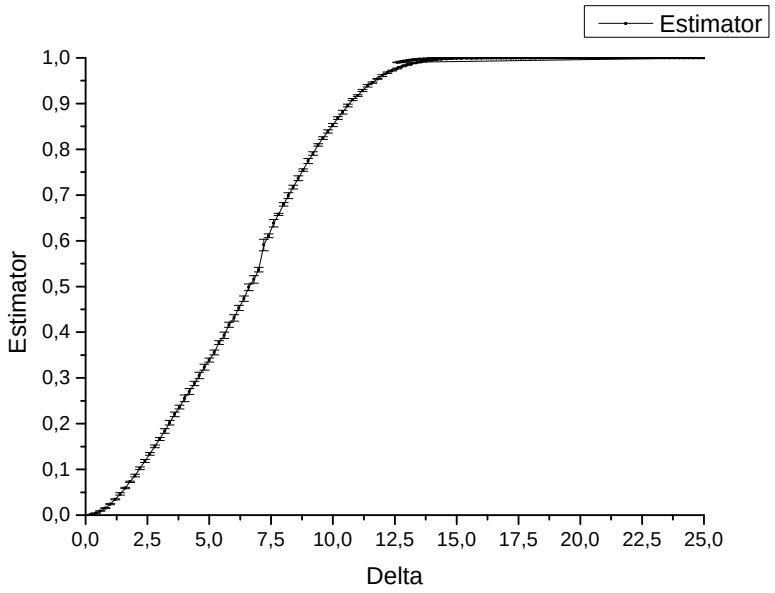

Figure 3.13: Estimator for 2-dimensional gaussian shell for a delta-region between 0 and 25 from MCMC with 100000 . The algorithm was run 10 times to determine mean value and standard-deviation.

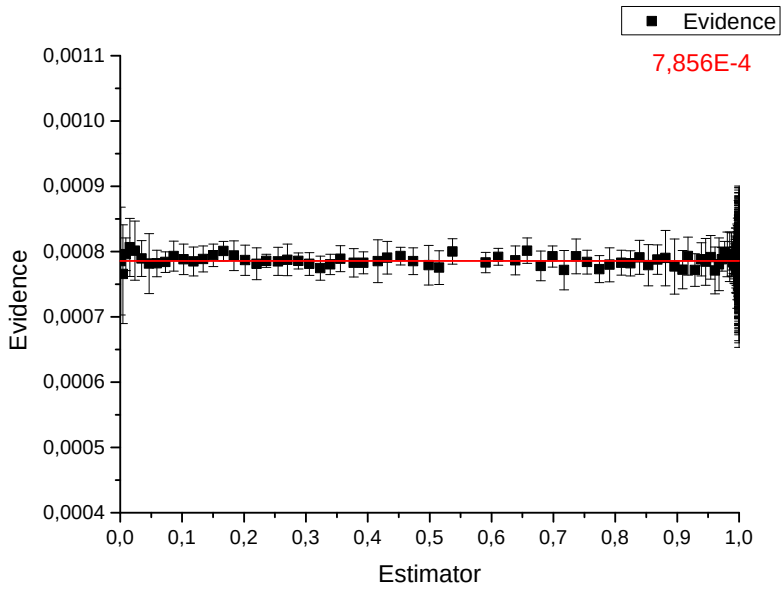

Figure 3.14: Estimator for 2-dimensional gaussian shell for a delta-region between 0 and 20 with a sample mean of 1000 samples. The algorithm was run 10 times to determine mean value and standard-deviation.

3.5 Gaussianshell in 10 dimensions

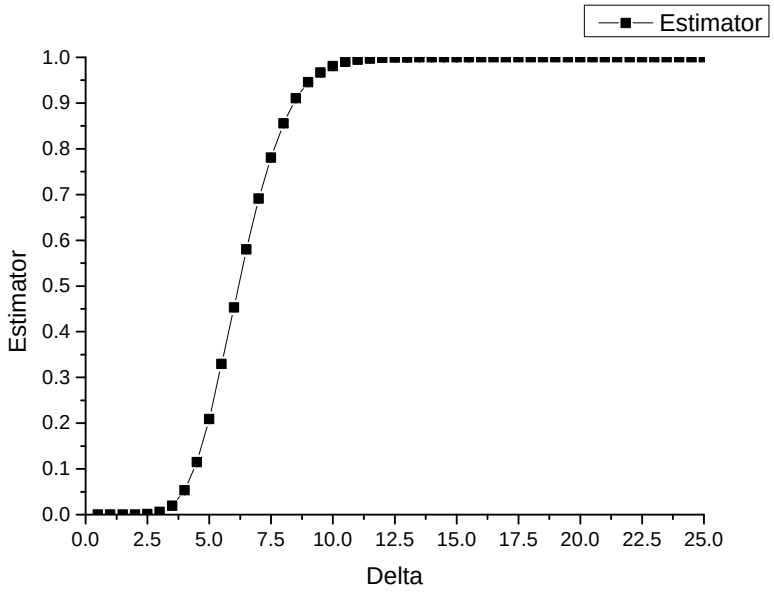

Figure 3.15: Estimator for 10-dimensional gaussian shell for a delta-region between 0 and 25 from MCMC with 100000 samples . The algorithm was run 10 times to determine mean value and standard-deviation.

To challenge our algorithm a little more, we repeat the gaussian shell calculation for 10 dimensions, we take the same expression for the evidence, (3.7), and take the same parameters radius $r = 5$, width $w = 2$ and the center point $c = (0, 0, 0, 0, 0, 0, 0, 0, 0, 0)$. The integral for 10 dimensions over a paramter space of [-100;100] gives as the result 1.081e-14, a sample mean calculation with as much as $1e + 8$ did not give any comparable results and varied strongly.
The results in figure 3.15 and 3.16 prove that, like for the gaussians, the high dimensional example is more interestingthis. While on the previous problems we found our evidence while estimator and delta-region only contributed to the precision, in this case we only find our evidence in the region where the estimator changes, for delta-values between 2.5 and 10. We see that this region als roughly corresponds with our parameters, width of 2.2 and diameter of 10,

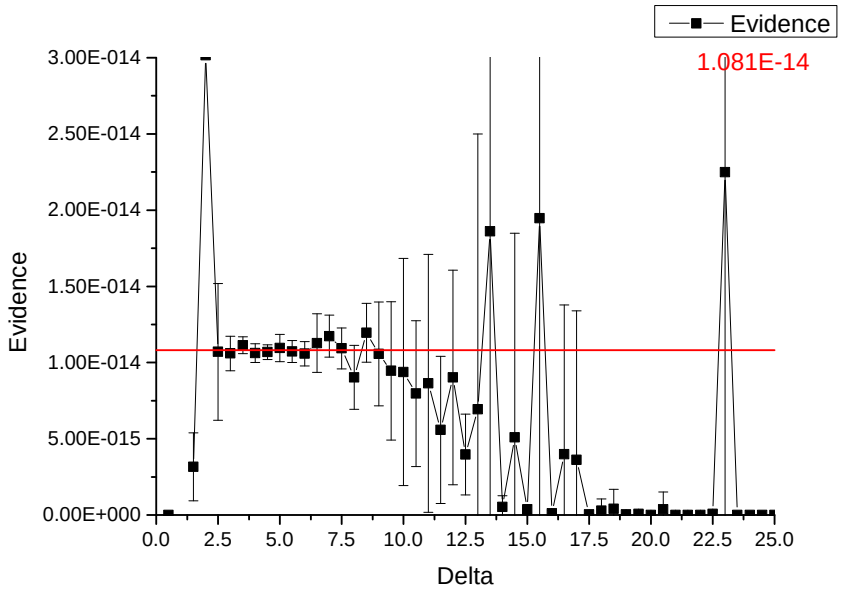

Figure 3.16: Evidence for 10-dimensional gaussian shell for a delta-region between 0 and 25 with a sample mean of 1000 samples. The algorithm was run 10 times to determine mean value and standard-deviation.

Chapter 4

Synopsis and outlook

In this thesis we showed a simple method for numerical integration, especially the integrals required to determine the so called evidence from the Bayesian Analysis. The presented method is based on post-processing of a MCMC-process, which can be easily provided by the BAT-Toolkit for example. The basic concept is to integrate an integral in a small area and estimate the whole integral through an analysis of the behaviour of our Markov chain.
We recognized that there are two conditions for optimal precision, a weak increase with an increasement of our estimator and a strong decrease with increasement of delta which mean the region. More interesting were the high dimensional problems, where we need an optimal region to determine our evidence values. The two 10-dimensional problems show, that even in this complex cases with prior regions of $200e^{10}$, the evidence can be calculated with a number of samples as small as 1000 if a MCMC was run successfully. The optimal region roughly corresponds with the region where our estimator is between 0 and 1 and we achieve good results even for very small estimators.
The next steps in transforming this method into a more generalized algorithm, that we can implement into the BAT-Toolkit, is to implement a method to find the region for unknown problems automatically. In this thesis we found this region by hand. We have noticed, that the region corresponds to the region where the estimator is between 0 and 1, which is useful because we can use this to find our region even for unknown functions or functions which cannot be visualized.
A starting point for this would be for example to take the estimate of the standard-deviation $\sqrt{<\mu^2> - <\mu>^2}$ of our MCMC-states μ to find a region where the precision of our evidence calculation is optimal, that means that at the same time the ratio of samples from MCMC in our region is big enough and the region is small enough to integrate it easily.

Appendix A

Analytical expression for the evidence of 1-dimensional Gaussian

The likelihood function for this problem is a product of Gaussians

$$P(D|\mu\sigma) = \prod_{i=1}^{d} \frac{1}{\sqrt{2\pi}\sigma} exp - \frac{(x_i - \mu)^2}{2\sigma^2} \tag{A.1}$$

so our evidence is

$$Z = \int \int P(D|\mu,\sigma) P_0(\mu,\sigma) d\mu d\sigma \tag{A.2}$$

using a flat prior $P_0(\mu,\sigma) = \frac{1}{\Delta\mu\Delta\sigma}$ we can rewrite this integral as

$$Z = \frac{1}{\Delta\mu\Delta\sigma(2\pi)^{\frac{d}{2}}} \int \frac{d\sigma}{\sigma^d} \int d\mu \prod_{i=1}^{d} exp - \frac{(x_i - \mu)^2}{2\sigma^2} \tag{A.3}$$

we will start now to evaluate the last integral $\int d\mu \prod_{i=1}^{d} exp - \frac{(x_i-\mu)^2}{2\sigma^2} = I$. Rewriting I we get:

$$\int d\mu e^{-\frac{S_{x^2} + 2S_x\mu - d\mu^2}{2\sigma^2}} \tag{A.4}$$

with $S_{x^2} = \sum_{i=1}^{d} x_i^2$ and $S_x = \sum_{i=1}^{d} x_i$, d the number of dimensions which is the number of datasets. Here we can extract the part of the exponential independent from μ. We define $a = \frac{N}{2\sigma^2}$, $b = \frac{S_x}{\sigma^2}$ and get the remaining exponent $-a\mu^2 + b\mu$. A substitution gives then

$$I = \sqrt{\frac{\pi}{4a}} \exp - \frac{S_{x^2}}{2\sigma^2} \exp \frac{b^2}{4a} [erf \frac{2a\mu - b}{2\sqrt{a}}] \tag{A.5}$$

for a large μ as we ours([-100;100]) the errorfunction gets simply 2 and we get

$$I = \sqrt{\frac{2\pi}{d}} \sigma \exp - \frac{dS_{x^2} + S_x^2}{2d\sigma^2} \tag{A.6}$$

Our full remaining integral is then

$$Z = \frac{1}{\Delta\mu\Delta\sigma(2\pi)^{\frac{d-1}{2}}\sqrt{d}} \int \frac{d\sigma}{\sigma^{d-1}} exp - \frac{dS_{x^2} + S_x^2}{2d\sigma^2} \tag{A.7}$$

with d again the number of datasets. Now we define $k = \frac{dS_{x^2}+S_x^2}{2d}$ so the integral is written as

$$\int \frac{d\sigma}{\sigma^{d-1}} \exp - \frac{k}{\sigma^2} = \frac{1}{2} \frac{(\frac{N}{2}-2)!}{k^{\frac{N}{2}-1}} \tag{A.8}$$

for $\frac{1}{\sigma^2} \leq \infty$ which is valid because our σ parameter space is [0.1;10]. Finally we get the expression we wanted:

$$Z = \frac{1}{\Delta_\mu \Delta_\sigma (2\pi)^{\frac{d-1}{2}} d^{\frac{1}{2}}} \frac{(\frac{d}{2}-2)!}{2(\frac{dS_{x^2}-S_x^2}{2d})^{\frac{d}{2}-1}} \tag{A.9}$$

Appendix B

Evidence for the single parameter Gaussians

We start again with our likelihood function which is for k= number of spacial dimensions and N number of Data-sets

$$P(D|\mu,\sigma) = \prod_{i=1}^{N} \frac{1}{(2\pi)^{\frac{N}{2}} |\Sigma|^{\frac{a}{2}}} e^{-\frac{1}{2}(x_i-\mu)^T \Sigma^{-1}(x_i-\mu)} \tag{B.1}$$

μ is the k-dimensional vector of our expectation value Σ is the covariance matrix, which is, in our case, a simple k-dimensional identity matrix. The integral for Z is

$$Z = \frac{1}{\Delta_\mu^k} \int \prod_{i=1}^{N} \frac{1}{(2\pi)^{\frac{N}{2}}} e^{-\frac{1}{2}(x_i-\mu)^T (x_i-\mu)} d\mu \tag{B.2}$$

As in appendix A we rewrite this integral using $S_{x^2} = \sum_{i=1}^{d} x_i^2$ and $S_x = \sum_{i=1}^{d} x_i$.

$$= \frac{1}{\Delta_\mu^k} \frac{1}{(2\pi)^{\frac{N}{2}}} \prod_{j=1}^{k} \int e^{-\frac{1}{2} S_{x_{ij}^2} - 2S_{x_{ij}}\mu_j + N\mu_j{}^2} d\mu_j \tag{B.3}$$

The integral in (B.3) is also the same as in A.

$$I = \sqrt{\frac{2\pi}{N}} \exp - \frac{N S_{x_{ij}{}^2} + S_{x_{ij}}{}^2}{2N} \tag{B.4}$$

To make this more plausible we express this through the variance of our data:

$$Var = \frac{1}{N} S_{x_{ij}{}^2} - \frac{1}{N^2} S_{x_{ij}}{}^2 \tag{B.5}$$

and finally get the expression for the evidence:

$$Z = \frac{1}{(\Delta_\mu^2 N)^{\frac{k}{2}}} \frac{1}{(2\pi)^{\frac{(N-1)k}{2}}} e^{-\frac{N}{2}(\sum_{j=1}^{k} Var[x_j])} \tag{B.6}$$

Appendix C

Code

C.1 1-dimensional gaussian without variance

```
#include <iostream>
#include <stdio.h>
#include <time.h>
#include <stdlib.h>//srandtime
#include <vector>
#include <cmath>//absvalue
#include <fstream>
using namespace std;

double Gauss(double tvar, double texp, double arg)
    {return (1/(tvar*sqrt(2*M_PI)))*exp(-0.5*pow(((arg-texp)/tvar),2));}

double Ran(double n)
    { return ((2*n*rand()+1.0)/(RAND_MAX+1.0)-n);}

double Metropolis(double tvar, double texp,vector<double> data,double step)//D
        {double temp=texp+Ran(step);
        double U=((rand()+1.0)/(RAND_MAX+1.0));
        if((temp<-100)||(temp>100))return texp;
        double R (1);
        for (int i=0;i<20;i++)
        R*=Gauss(tvar,temp,data[i])/Gauss(tvar,texp,data[i]);
        if (R>U) return temp;
        else return texp;}

   double Compare(double ein,double ein2,vector<double> data)//Algorithm used
        {double R (1);
        for (int i=0;i<20;i++)
        {R*=Gauss(1,ein,data[i])/Gauss(1,ein2,data[i]);}
```

```
30          if (R>1) return ein;
31          else     return ein2;}
32
33      double Smean(double counts, double range,vector<double> data, double t
34          {double sum (0);
35          for (int i=0;i<counts;i++)
36                  {double y=tmode+Ran(range);
37                  double temp1 (1);
38                  for (int j =0;j<20;j++)  temp1*=Gauss(tvar,y,data[j]);
39                  sum+=temp1;
40              }return sum/counts;}
41
42
43
44  int main()
45  {srand(time(NULL));
46  ifstream ifs("Data1.txt");//file to load Data from
47  vector<double> Data (20);//save Data
48  double mode (10);
49  double x=Ran(20);
50  double iterations (100000);//number of MCMC states produced
51  vector<double> save (iterations);//save MCMC states
52
53  for (int i=0;i<20;i++)//Loading Data
54      {ifs>>value;
55      Data[i]=value;}
56      ifs.close();
57
58  {for (int j=0;j<iterations;j++)
59     {x=Metropolis(1,x,Data,5);
60      save[j]=x;
61      mode=Compare(mode,x,Data);}
62
63
64  double counter (0);
65  for (int j=0;j<iterations;j++){if (abs(save[j]-mode)<Delta){counter+=1;}}
66  double Faktor=(Delta/100.0)*(double)iterations/(double)counter;
67  double RSample=Faktor*Smean(10000,Delta,Data,1,mode);
68  double result (4.95679e-18);//Analytical Integral
69  double result2 (5.56649e-18);//Sample Mean Integration
70  }
```

C.2 1-dimensional gaussian with variance

C.2.1 Data iteration

```
#include <iostream>
#include <stdio.h>
#include <time.h>
#include <stdlib.h>//srandtime
#include <vector>
#include <cmath>//absvalue
#include <fstream>
using namespace std;

double Gauss(pair<double,double> ein, double arg)
    {return (1/(ein.second*sqrt(2*M_PI)))*exp(-0.5*pow(((arg-ein.first)/ein.se


double Ran(double n)//create random value between -n and n
    { return ((2*n*rand()+1.0)/(RAND_MAX+1.0)-n);}

double Metropolis(pair<double,double> ein, double arg,double step)//Metropolis
        {double temp=arg+Ran(step);
        if ((temp<(-100) || temp>100){return arg}
        double U=((rand()+1.0)/(RAND_MAX+1.0));
        double R =Gauss(ein,temp)/Gauss(ein,arg);
        if (R>U) return temp;
        else return arg;}

int main()
{srand(time(NULL));
ofstream ofs("Data1.txt");//file to save Data to
pair<double,double> Parameter (0,5);
double x=Ran(10);
Parameter.first=0;
Parameter.second=1;
for (int i=0;i<40;i++){x= Metropolis(Parameter,x,10);}//Prerun for MCMC
for (int i=0;i<50;i++){x=Metropolis(Parameter,x,5);//Data Creation
                       ofs<<x<<endl;}
ofs.close();}
```

C.2.2 Parameter variation and sample mean integration

```
#include <iostream>
#include <stdio.h>
#include <time.h>
#include <stdlib.h>//srandtime
#include <vector>
#include <cmath>//absvalue
#include <fstream>//ifstream, ofstream
using namespace std;

//The Gauss function for 1 Dimension with 2 parameters
double Gauss(double tvar, double texp, double arg)
    {return (1/(tvar*sqrt(2*M_PI)))*exp(-0.5*pow(((arg-texp)/tvar),2));}

//create random value from -n to n
double Ran(double n)
    { return ((2*n*rand()+1.0)/(RAND_MAX+1.0)-n);}
//Metropolis for Parameter iteration
pair<double,double> Metropolis(pair<double,double> ein,vector<double> data
        {pair<double,double> temp;
        temp.first=ein.first+Ran(step);
        temp.second=(ein.second +Ran(step2));
        double U=((rand()+1.0)/(RAND_MAX+1.0));
        double R (1);
        if ((temp.first <(-100) || temp.first >100)||(temp.second<0.1 || tem
        else
        {for (int i=0;i<50;i++)
        {R*=Gauss(temp,data[i])/Gauss(ein,data[i]);}
        if (R>U) {return temp;}
        else return ein;}}


//method to find mode from MCMC
pair<double,double> Compare(pair<double,double> ein,pair<double,double> ei
        {double R (1);
        for (int i=0;i<50;i++)
        {R*=Gauss(ein,data[i])/Gauss(ein2,data[i]);}
        if (R>1) return ein;
        else    return ein2;}

int main()
{srand(time(NULL));
```

```
42 ifstream ifs("Data1.txt");//open file to load data from
43 int Dataset (50);//give number of data
44 vector<double> Data (Dataset);//vector to save data
45 double value (0);
46 for (int i=0;i<Dataset;i++)//Loading Data
47     {ifs>>value;
48     Data[i]=value;}
49     ifs.close();
50 int iterations =100000;//Number of iterations for the Markov chain
51 pair<double,double> Parameter (0,5);
52 pair<double,double> mode (5,5);
53
54 vector<pair<double,double> > save (iterations);//vector to save MCMC-states
55
56 {double Delta ((k)/2.5);
57 //Parameter Initialisation
58 Parameter.first=Ran(100);
59 Parameter.second=abs(Ran(10));
60
61 Parameter Iteration
62 for (int i=0;i<iterations;i++){Parameter=Metropolis(Parameter,Data,2,1);
63                                save[i].first=Parameter.first;
64                                save[i].second=Parameter.second;
65                                mode=Compare(Parameter,mode,Data);}
66
67
68 // Determination of number of states in region
69 double counter(0);
70 for (int i=0;i<iterations;i++)if ((abs(save[i].first-mode.first)<Delta)&&(abs(
71
72 //Sample Mean over reduced region;
73 double integral (0);
74 double count2 (1000);
75 //Finding the variance intervall
76 double point1=mode.second/(pow(10,Delta/100));
77 double point2=mode.second*pow(10,(Delta/100));
78 double delta2=(point2-point1)/2;
79 //Sample Mean Random values
80 for (int i=0;i<count2;i++)
81 {pair<double,double> var;
82 var.first=(mode.first +Ran(Delta));
```

```
var.second=(point1+delta2+Ran(delta2));
double temp (1);
for (int i=0;i<Dataset;i++)temp*=Gauss(var,Data[i]);//Likelihood function
integral+=temp;}
double Volumen=4*Delta*delta2;
double Volumen2=200*9.9;
double evidence=((1/count2)*integral*(Volumen/Volumen2)*(iterations/counte
double result (1.08606e-25);}//Integral in Wolframalpha;
```

C.3 10-dimensional gaussian

C.3.1 Data generation

```
#include <iostream>
#include <stdio.h>
#include <time.h>
#include <stdlib.h>//srandtime
#include <vector>
#include <cmath>//absvalue
#include <fstream>
using namespace std;


double Gauss(vector<double> texp,vector<double> arg)
        {double var (0);
        double erw (0);
        for (int i=0;i<10;i++)
        {erw+=pow(texp[i]-arg[i],2); }
        double temperw=sqrt(erw);
        return(1/(pow((2*M_PI),5))*exp(-0.5*pow((temperw),2)));}

double Ran(double n)
    { return ((2*n*rand()+1.0)/(RAND_MAX+1.0)-n);}

vector<double> Metropolis( vector<double> texp,vector<double> arg,double s
        {vector<double> temp(10);
        for (int i=0;i<10;i++) temp[i]=(arg[i]+Ran(step));
        double U=((rand()+1.0)/(RAND_MAX+1.0));
        double R =Gauss(texp,temp)/Gauss(texp,arg);
        if (R>U) return temp;
        else return arg;}
```

```
29
30
31 int main()
32 {srand(time(NULL));
33 ofstream ofs("MDData1.txt");
34 vector<double> expectation(10);
35 vector<double> variance(10);
36 vector<double> x (10);
37 for (int i=0;i<10;i++)
38     {expectation[i]=0;
39     variance[i]=1;
40     x[i]=Ran(2);}
41 for (int i=0;i<200;i++)x=Metropolis(expectation,x,1);
42 for (int i=0;i<20;i++)
43     {x=Metropolis(expectation,x,1);
44     for (int j=0;j<10;j++)ofs<<x[j]<<"   ";
45     ofs<<endl;}
46     ofs.close();}
```

C.3.2 Evidence calculation

```
1 #include <iostream>
2 #include <stdio.h>
3 #include <time.h>
4 #include <stdlib.h>//srandtime
5 #include <vector>
6 #include <cmath>//absvalue
7 #include <fstream>
8 using namespace std;
9
10 int Dataset (10);//number of Datasets for our evidence
11
12 double Gauss(vector<double> texp,vector<double> arg)
13         {double var (0);
14         double erw (0);
15         for (int i=0;i<10;i++)
16         {erw+=pow(texp[i]-arg[i],2); }
17         double temperw=sqrt(erw);
18         return(1/(pow((2*M_PI),5))*exp(-0.5*pow((temperw),2)));}
19
20 double Ran(double n)
```

```
21      { return ((2*n*rand()+1.0)/(RAND_MAX+1.0)-n);}
22
23  vector<double> Metropolis(vector<double> texp,vector<vector<double> >data,
24          {vector<double> temp(10);
25          for (int i=0;i<10;i++) {temp[i]=(texp[i]+Ran(step));
26          if (abs(temp[i]>100)) return texp;}
27          double U=((rand()+1.0)/(RAND_MAX+1.0));
28          double R (1);
29          for (int i=0;i<Dataset;i++)R*=Gauss(temp,data[i])/Gauss(texp,data[
30          if (R>U) return temp;
31          else return texp;}
32
33     vector<double> Compare(vector<double> ein,vector<double> ein2,vector<ve
34          {double R (1);
35          for (int i=0;i<Dataset;i++)
36          {R*=Gauss(ein,data[i])/Gauss(ein2,data[i]);}
37          if (R>1) return ein;
38          else     return ein2;}
39
40  double Smean(double counts, double range,vector<vector<double> > data,vect
41          {double sum (0);
42          vector<double> y (10);
43          for (int i=0;i<counts;i++)
44                  {for (int j=0;j<10;j++) y[j]=tmode[j]+Ran(range);
45                  double temp1 (1);
46                  for (int j =0;j<Dataset;j++)  temp1*=Gauss(y,data[j]);
47                  sum+=temp1;}
48                  return sum/counts;}
49
50  int main()
51  {srand(time(NULL));
52      double iterations =100000;
53      vector<vector<double> > Data(Dataset,vector<double> (10));
54      ifstream ifs("MDData1.txt");//reading the Data
55      vector<double> value (10);
56      for (int i=0;i<Dataset;i++)
57          {for (int j=0;j<10;j++){ifs>>value[j];}
58          Data[i]=value;}
59      ifs.close();
60
61
```

```
62
63 vector<double> mode (10,10);
64 vector<double> vexp (10,0);
65 vector<vector<double> > save(iterations,vector<double> (10,0));
66 for (int i=0;i<10;i++){vexp[i]=Ran(2);}
67 for (int i=0;i<iterations;i++){vexp=Metropolis(vexp,Data,1);
68                                save[i]=vexp;
69                                mode=Compare(vexp,mode,Data);}
70
71 double Delta (1);//Defining our region
72 double counter (0);
73 for (int i=0;i<iterations;i++){vexp=Metropolis(vexp,Data,1);
74                                save[i]=vexp;
75                                mode=Compare(vexp,mode,Data);}
76
77 for (int i=0;i<iterations;i++)//calculating the estimator in 10 dimensions
78 {double subcounter (0);
79 for (int j=0;j<10;j++)
80 {if (abs(save[i][j]-mode[j])<Delta)subcounter+=1;}
81 if (subcounter==10){counter+=1;}
82 }
83 double Volumen1=pow(2*Delta,10);
84 double Volumen2=pow(200,10);
85 double Rintegral=Smean(1000,Delta,Data,mode);
86 double Faktor=(iterations/counter)*(Volumen1/Volumen2);
87 cout<<"Result:  "<<Rintegral*Faktor<<endl;}//Result
```

C.4 2-dimensional shell

```
1 #include <iostream>
2 #include <stdio.h>
3 #include <time.h>
4 #include <stdlib.h>//srandtime
5 #include <vector>
6 #include <cmath>//absvalue
7 #include <fstream>
8 using namespace std;
9
10 double Shell(double r, double w, vector<double> c,vector<double> arg)
11 {   double temp (0);
12     for (int i=0;i<2;i++){temp+=pow(c[i]-arg[i],2);}
```

```
13      return 1/(w*sqrt(2*M_PI))*exp(-0.5*pow(((sqrt(temp)-r)/w),2));
14  }
15  double Ran(double n)
16      { return ((2*n*rand()+1.0)/(RAND_MAX+1.0)-n);}
17
18  vector<double> Metropolis(vector<double> tpos, vector<double> tcentre, dou
19      {vector<double> temp(2);
20      for (int i=0;i<2;i++)temp[i]=tpos[i]+Ran(step);
21      if ((temp[0]<-100 || temp[0]>100)||(temp[1]<-100 || temp[1]>100))retur
22      double U=((rand()+1.0)/(RAND_MAX+1.0));
23      double R =Shell(5,2,tcentre,temp)/Shell(5,2,tcentre,tpos);
24      if (R>U) return temp;
25          else return tpos;}
26
27  vector<double> compare(vector<double> tcentre,vector<double> mode,vector<d
28                  {if (Shell(tr,tw,tcentre,param)>Shell(tr,tw,tcentre,mode))
29                  else return mode;}
30
31
32  int main()
33  {srand(time(NULL));
34  int iterations =100000;
35  cout<<"Iterationen:"<<iterations<<endl;
36  vector<vector<double> > Data(iterations,vector<double> (2));
37  vector<double> Mode (2);
38  vector<vector<double > > save (iterations,vector<double> (2));
39
40  //Create Starting Parameters
41  vector<double> Centre (2,0);
42  vector<double> Position(2);
43  double width (2);
44  double radius (5);
45  Position[0]=Ran(5);
46  Position[1]=Ran(5);
47
48
49  double Delta =((k+1))/10;
50  //Data Iteration
51  for (int i=0;i<100;i++)Position= Metropolis(Position,Centre,10);
52  for (int i=0;i<iterations;i++){ Position=Metropolis(Position,Centre,4);
53                                  Mode=compare(Centre,Mode,Position,radius,w
```

```
54                                      Data[i]=Position;}
55 // find number of samples in area
56 double counter (0);
57 for (int i=0;i<iterations;i++){if ((abs(Data[i][0]-Mode[0])<Delta) &&(abs(Data
58
59 // Evidence
60 double RIntegral (0);
61 double counter2 (1000);
62 double Volumen=4*Delta*Delta;
63 double Volumen2=200*200;
64 for(int i=0;i<counter2;i++)
65 {vector<double> temp (2);
66 temp[0]=Mode[0]+Ran(Delta);
67 temp[1]=Mode[1]+Ran(Delta);
68 RIntegral+=Shell(radius,width,Centre,temp);
69 }
70 double result (0.000785643);//"True value determined with sample mean over 100
71 double Faktor =Volumen/Volumen2*iterations/counter*1/counter2;
```

C.5 10-d Gaussian shell

```
1 #include <iostream>
2 #include <stdio.h>
3 #include <time.h>
4 #include <stdlib.h>//srandtime
5 #include <vector>
6 #include <cmath>//absvalue
7 using namespace std;
8
9 double Shell(double r, double w, vector<double> c,vector<double> arg)
10 {   double temp (0);
11     for (int i=0;i<10;i++){temp+=pow(c[i]-arg[i],2);}
12     return 1/(w*sqrt(2*M_PI))*exp(-0.5*pow(((sqrt(temp)-r)/w),2));}
13
14 double Ran(double n)
15     { return ((2*n*rand()+1.0)/(RAND_MAX+1.0)-n);}
16
17 vector<double> Metropolis(vector<double> tpos, vector<double> tcentre, double
18     {vector<double> temp(10);
19     for (int i=0;i<10;i++)temp[i]=tpos[i]+Ran(step);
20     for (int j=0;j<10;j++)
```

```
21      {if (temp[j]<-100 || temp[j]>100)return tpos;}
22      double U=((rand()+1.0)/(RAND_MAX+1.0));
23      double R =Shell(5,2,tcentre,temp)/Shell(5,2,tcentre,tpos);
24      if (R>U) return temp;
25          else return tpos;}
26
27  vector<double> compare(vector<double> tcentre,vector<double> mode,vector<d
28                  {if (Shell(tr,tw,tcentre,param)>Shell(tr,tw,tcentre,mode))
29                  else return mode;}
30
31  int main()
32  {srand(time(NULL));
33  int iterations =100000;
34  vector<vector<double> > Data(iterations,vector<double> (10));
35  vector<double> Mode (10,10);
36  vector<vector<double > > save (iterations,vector<double> (10));
37
38  //Create Starting Parameters
39  vector<double> Centre (10,0);
40  vector<double> Position(10);
41  double width (2);
42  double radius (5);
43  for (int i=0;i<10;i++)  Position[i]=Ran(10);
44
45
46  //Data Iteration
47  for (int i=0;i<100;i++)Position= Metropolis(Position,Centre,10);
48  for (int i=0;i<iterations;i++){Position=Metropolis(Position,Centre,4);
49                                 Mode=compare(Centre,Mode,Position,radius,w
50                                 Data[i]=Position;}
51
52  double counter (0);
53  double Delta (5);
54  for (int i=0;i<iterations;i++)
55  {   double subcounter (0);
56      for(int j=0;j<10;j++){if ((abs(Data[i][j]-Mode[j])<Delta)) subcounter+
57      if (subcounter==10){counter+=1;}}
58
59  //Reduced Evidence
60  double RIntegral (0);
61  double counter2 (100000);
```

```
62 double Volumen=pow(2.0*Delta,10);
63 double Volumen2=pow(200,10);
64 for(int i=0;i<counter2;i++)
65 {vector<double> temp (10);
66 for (int j=0;j<10;j++)  temp[j]=Mode[j]+Ran(Delta);
67 RIntegral+=Shell(radius,width,Centre,temp);}
68 double Faktor =Volumen/Volumen2*iterations/counter*1/counter2;
69
70 double result (0.000784028);//Sample mean 10000000;
71 double result2(1.08e-14);//Wolframalpha Integral probably wrong area;}
```

Appendix D

Data

D.1 1-dimensional gaussian with variance

```
1  0.508759
2  0.508759
3  0.508759
4  0.602478
5  0.602478
6  0.602478
7  0.602478
8  −0.219635
9  −0.219635
10 −0.219635
11 1.0777
12 1.0777
13 0.0773621
14 0.0773621
15 0.0773621
16 0.0773621
17 0.0773621
18 0.0773621
19 0.0773621
20 0.0773621
21 0.0773621
22 0.0773621
23 0.0773621
24 0.0773621
25 −0.0370483
26 −0.0370483
27 −0.0370483
28 −0.0370483
29 −0.0370483
```

30 −0.0370483
31 −1.2269
32 −1.2269
33 −1.2269
34 −1.2269
35 −1.05322
36 0.974701
37 0.974701
38 0.974701
39 0.302734
40 0.498383
41 0.498383
42 −0.428711
43 −0.191864
44 −0.191864
45 −0.191864
46 −0.191864
47 −0.191864
48 −0.841553
49 −1.09665
50 −1.09665

D.2 1-dimensional gaussian without variance

1 −1.24298
2 1.59134
3 2.13007
4 1.63406
5 1.13513
6 0.504181
7 0.504181
8 −0.453064
9 −0.95108
10 0.650757
11 −1.34891
12 −0.0253906
13 1.01523
14 −0.5979
15 −0.5065
16 −1.7796
17 −0.757294

18 −0.757294
19 −0.407166
20 −2.95432

D.3 10-dimensional gaussian

Here the data consists of 10-dimensional vectors, 1 vector per line.

1 0.0970154 0.146332 1.00449 1.2128 0.837555 1.41959 0.682465 −0.269806 0.489349 1.27505
2 0.0970154 0.146332 1.00449 1.2128 0.837555 1.41959 0.682465 −0.269806 0.489349 1.27505
3 0.0970154 0.146332 1.00449 1.2128 0.837555 1.41959 0.682465 −0.269806 0.489349 1.27505
4 0.0970154 0.146332 1.00449 1.2128 0.837555 1.41959 0.682465 −0.269806 0.489349 1.27505
5 0.0119019 0.42157 0.604309 1.28131 0.195068 1.75763 −0.261658 0.0476685 0.595825 0.835266
6 0.611664 −0.502838 0.329071 1.12436 −0.40213 1.40079 −1.08725 −0.161102 0.00125122 0.714081
7 0.611664 −0.502838 0.329071 1.12436 −0.40213 1.40079 −1.08725 −0.161102 0.00125122 0.714081
8 0.123596 0.447083 1.32294 0.310242 0.0438843 1.13495 −1.1546 0.181335 −0.484558 0.258728
9 0.123596 0.447083 1.32294 0.310242 0.0438843 1.13495 −1.1546 0.181335 −0.484558 0.258728
10 0.123596 0.447083 1.32294 0.310242 0.0438843 1.13495 −1.1546 0.181335 −0.484558 0.258728}

Bibliography

[Robert,Casella,2004] "Monte Carlo Statistical Methods",Second Edition,New York

[Beaujean,Caldwell, 2013] "Initializing adaptive importance sampling with Markov chains", arxiv1304.7808v1

[Caldwell,Kollar,Kröninger] "BAT-The Bayesian Analysis Toolkit",Computer Physics Communications 180(2009)p.2197-2209

[Chib,Jeliazkov] "Marginal Likelihood from the Metropolis-Hastings Output", Journal of the American Statistical Association 96 (2001) 270

[Skilling] "Nested Sampling for the General Bayesian Computation",Bayesian Analysis 1 (2006) 833