Gravitational wave detection and data analysis for pulsar timing arrays

Haasteren, R. van

5 Marginal likelihood calculation with MCMC methods

The most important kind of freedom is to be what you really are. [. . .] There can't be any large-scale revolution until there's a personal revolution, on an individual level. It's got to happen inside first.

Jim Morrison

Abstract

Markov Chain Monte Carlo (MCMC) methods have revolutionised Bayesian data analysis over the years by making the direct computation of posterior probability densities feasible on modern workstations. However, the calculation of the prior predictive, the marginal likelihood, has proved to be notoriously difficult with standard techniques. In this chapter a method is presented that lets one calculate the marginal likelihood using nothing but the results from standard MCMC algorithms, like Metropolis-Hastings. This new method is compared to other methods like nested sampling, and outperforms the latter in some cases. One of the toy problems considered in this chapter is the analysis of mock pulsar timing data, as encountered in pulsar timing array projects. This method is expected to be useful as well in other problems in astrophysics, cosmology and particle physics.

To be submitted to MNRAS

5.1 Introduction

Bayesian inference has proved over the years to be a very powerful tool in many branches of science as it gives a very clear prescription of how to analyse datasets without loss of information, even for very complicated models. On the practical side, in performing Bayesian analysis two difficult problems often emerge:

1. Producing the posterior probability density functions (PDFs) for several interesting parameters requires marginalisation, i.e. the integration of the full joint posterior probability distribution function (PDF) over most model parameters. In a majority of cases this must be done numerically, and it is common to employ a Markov Chain Monte-Carlo (MCMC) algorithm for performing the integration.
2. Model selection requires the calculation of the Bayes factor: the ratio of the marginal likelihood (ML) of two competing models. This marginal likelihood, sometimes also called the evidence, is the normalisation constant required to have the likelihood times the prior PDF (when normalised called the posterior PDF) integrate to unity when integrating over all parameters. The calculation of this value can be notoriously difficult using standard techniques.

The use of Markov Chain Monte Carlo (MCMC) algorithms, such as the Metropolis-Hastings algorithm, has become extremely popular in the calculation of the marginal posterior PDFs. MCMC algorithms allow one to sample from posterior distribution of complicated statistical models, greatly reducing the effort involved in evaluating numerical integrals. MCMC algorithms typically sample from a posterior distribution in a way that does not require explicit normalization. This is a virtue when only one model is of interest and only parameter estimation must be pursued. But it means that most MCMC algorithms do not directly address problem (2).

Over the years, several methods capable of calculating the ML have been developed (including Newton & Raftery, 1994; Earl & Deem, 2005; Skilling, 2004; Feroz et al., 2009). Typically, the technical challenges of producing the posterior PDFs, and the ML, that these methods try to overcome can be divided in three stages:
1. Finding the high probability density (HPD) regions of the posterior in parameter space.
2. Maximising the posterior to find the best estimators.
3. Sampling the HPD regions as accurately as possible.
It is very unlikely that one algorithm will outperform all others in all three aspects, and complicated problems may require a combination of algorithms to be efficient. Results of ML calculations can rarely be checked for sanity without additional major effort, usually involving running the same or a different algorithm for the same problem to verify a result. Especially for integrals in very high dimensional spaces, which are particularly challenging, crosschecks between algorithms may be desired.

In this chapter we present a method to calculate the ML from the MCMC chains of regular MCMC algorithms. This method can be applied to chains that have already been run, provided that the values of the likelihood times the prior have been saved together with the values of the parameters at each point in the chain.

By using MCMC chains, the method is especially efficient in stage (3), sampling the HPD region accurately if found correctly. In problems where sampling the HPD region efficiently is the greatest challenge, this new method could be of particular interest. The error on the ML can be calculated using a batch means procedure.

The outline of the chapter is as follows. In Section 5.2 we briefly review the basic aspects of Bayesian inference for parameter estimation and model selection. Then we provide the reader with some necessary details on MCMC in Section 5.3, where we also outline the new algorithm to calculate the ML. In Section 5.4 we assess the strengths and weaknesses of the new method relative to those that use nested sampling or parallel tempering. In Section 5.5 we test all competing algorithms on some toy problems like the analysis of pulsar timing data as encountered in pulsar timing array projects.

5.2 Bayesian inference

Bayesian inference methods provide a clear and consistent approach to parameter estimation and model selection. Consider a model, or Hypothesis, H with parameters $\vec{\Theta}$ for a dataset $\vec{d}$. Then Bayes' theorem states that

$$P\left(\vec{\Theta} \mid \vec{d}, H\right) = \frac{P\left(\vec{d} \mid \vec{\Theta}, H\right) P\left(\vec{\Theta} \mid H\right)}{P\left(\vec{d} \mid H\right)}, \tag{5.1}$$

where $P(\vec{\Theta}) := P(\vec{\Theta} \mid \vec{d}, H)$ is the posterior PDF of the parameters, $L(\vec{\Theta}) := P(\vec{d} \mid \vec{\Theta}, H)$ is the likelihood function, $\pi(\vec{\Theta}) := P(\vec{\Theta} \mid H)$ is the prior, and $z := P(\vec{d} \mid H)$ is the marginal likelihood.

The ML is the factor required to normalise the posterior over $\vec{\Theta}$:

$$z = \int L\left(\vec{\Theta}\right) \pi\left(\vec{\Theta}\right) \mathrm{d}^m\Theta, \tag{5.2}$$

where m is the dimensionality of $\vec{\Theta}$. The ML can then be used to calculate the so-called odds ratio in favor of model H_1 over H_0, which allows one to perform model selection:

$$\frac{P\left(H_1 \mid \vec{d}\right)}{P\left(H_0 \mid \vec{d}\right)} = \frac{z_1}{z_0} \frac{P\left(H_1\right)}{P\left(H_0\right)}, \tag{5.3}$$

where $P(H_0)$ and $P(H_1)$ are the prior probabilities for the different models . As with the prior for a model parameters, the prior probability for a model should be chosen to reflect the available information, if any.

In parameter estimation problems, one is interested in the posterior PDF, marginalised over all nuisance parameters. In this case, knowing the marginalised likelihood is not required, since the resulting marginalised posterior PDF can be normalised after the integration. However, when model choice itself is uncertain, the ML is no longer an uninteresting normalisation constant, but a key quantity that allows us to perform model selection.

Being the average of the likelihood over the prior distribution, the ML is larger for a model if more of its parameter space is likely and smaller for a model with large areas in its parameter space having low likelihood values. Even if the likelihood function has high peaks, in order to increase the ML these peaks must compensate for the areas in its parameter space where the likelihood is low. Thus the ML automatically implements Occam's razor: a simpler theory with compact parameter space will have a larger ML than a more complicated model, unless the latter is significantly better at explaining the data.

5.3 Markov Chain Monte Carlo

Markov Chain Monte Carlo methods (MCMC) can be used to sample from very complicated, high dimensional distribution; for Bayesian inference it is usually the posterior PDF. The method presented in this chapter could be useful for integration problems other than ML calculation, so we use the more general $f(\vec{\Theta})$ to denote this unnormalised function. The samples drawn from a distribution proportional to this function can then be used to perform the integrals we need for the marginal posterior PDFs and, as we show, for the ML. The exact mechanism that produces these samples can differ between MCMC methods and is irrelevant for the purposes of this work, but the result is always a large number of samples distributed according to $f(\vec{\Theta})$. The main advantage of this is that we do not have to sample from the entire volume of the parameter space or the prior, but only from a small fraction of it: the high probability density (HPD) regions. Especially for functions in high-dimensional parameter spaces this feature is crucial for efficient ML evaluation.

In Section 5.3.2 we show that all information needed to calculate the ML is already present in the samples of the MCMC. Then we give a practical example of how to calculate the integral in practice in Section 5.3.4.

5.3.1 The harmonic mean estimator

A very simple ML estimator can be derived from Equation (5.1):

$$\begin{aligned}\frac{1}{z} &= \frac{P\left(\vec{\Theta} \mid \vec{d}, H\right)}{L\left(\vec{\Theta}\right)\pi\left(\vec{\Theta}\right)} = \int \frac{P\left(\vec{\Theta} \mid \vec{d}, H\right)}{L\left(\vec{\Theta}\right)} \mathrm{d}^m\Theta \\ &= \left\langle \frac{1}{L\left(\vec{\Theta}\right)} \right\rangle_P ,\end{aligned} \tag{5.4}$$

where we have multiplied the right hand side by $\pi(\vec{\Theta})$, and integrated over all parameters. The expectation is an expectation over the posterior PDF. Equation (5.4) is then used to form the harmonic mean (HM) estimator of the ML (Newton & Raftery, 1994):

$$\hat{z}_{HM} = \frac{1}{\frac{1}{N}\left(\sum_{i=1}^{N} \frac{1}{L(\vec{\Theta}_i)}\right)}, \tag{5.5}$$

where the summation is over all N MCMC samples.

Although the HM estimator follows directly from a true identity, it has some statistically undesirable properties. The HM estimator does converge to the correct value, but the variance of the estimator can be infinite. This is because the likelihood can reach values arbitrarily close to zero with nonzero probability in the case that these values are permitted by the prior. This is especially important if a peak in the likelihood is smaller than a corresponding peak in the prior PDF, i.e. when the data is informative. As a result, in the case that the data is informative, the HM estimator converges much slower than estimators that follow the central limit theorem (Wolpert, 2002).

5.3.2 The truncated harmonic mean estimator

In this chapter we advocate a modification to the HM estimator by eliminating the probability that we include samples in our estimator for which the likelihood gets arbitrarily close to zero. In the derivation in this section, we consider the general function $f(\vec{\Theta}) = zp(\vec{\Theta})$, where $p(\vec{\Theta})$ is a PDF proportional to $f(\vec{\Theta})$. We are interested in the integral over all parameters of $f(\vec{\Theta})$. Analogous to Equation (5.4), we now integrate $1/z$, but we now perform the integral only over a HPD region of $f(\vec{\Theta})$, given by $\mathcal{V} \subset \mathbb{R}^n$ with volume V. This gives:

$$\frac{V}{Z} = \int_{\mathcal{V}} \frac{p(\vec{\Theta})}{f(\vec{\Theta})} \mathrm{d}^m\Theta. \tag{5.6}$$

This equation can be used to form an estimator similar to the HM estimator as follows. First note that the PDF $p(\vec{\Theta})$ is not normalised on $\mathcal{V}$. By defining the PDF $p_{\mathcal{V}}(\vec{\Theta}) = wp(\vec{\Theta})$, we can write a truncated harmonic mean estimator for the ML:

$$\frac{V}{Z} = w \int_{\mathcal{V}} \frac{p_{\mathcal{V}}(\vec{\Theta})}{f(\vec{\Theta})} \mathrm{d}^m\Theta = w \left\langle \frac{1}{f(\vec{\Theta})} \right\rangle_{p_{\mathcal{V}}} . \tag{5.7}$$

Now, assuming that $N_{\mathcal{V}}$ samples of the total N MCMC samples are within $\mathcal{V}$, the HPD region of $f(\vec{\Theta})$, we can write down the new estimator:

$$\hat{z} = \frac{V}{\frac{N_{\mathcal{V}}}{N} \left(\frac{1}{N_{\mathcal{V}}} \sum_{\vec{\Theta}_i \in \mathcal{V}} \frac{1}{f(\vec{\Theta})} \right)}, \tag{5.8}$$

where $N_{\mathcal{V}}/N$ appears as an estimator for w. Equation (5.8) is the estimator we propose as an improvement to the HM estimator in this chapter.

5.3.3 Analogy with tessellations

In order to provide a more intuitive approach to the estimator presented in Section 5.3.2, we try to motivate the use of that estimator in an entirely different way in this section. Consider the function:

$$f(x, y) = \exp\left(-ax^2 - by^2\right), \tag{5.9}$$

where a and b are arbitrary model parameters. For the values $a = 1/5$ and $b = 2/5$, a Metropolis algorithm with $N = 40000$ samples yield a distribution of samples similar to Figure 5.1. We use the notation $\vec{\Theta}_i = (x_i, y_i)$ to indicate the i^{th} sample. We would now like to calculate the integral

$$z = \int \int f(x, y) \, \mathrm{d}x\mathrm{d}y. \tag{5.10}$$

In any MCMC algorithm, the function values $f(\vec{\Theta}_i)$ for each point are evaluated, and usually these values are stored together with the values of the parameters. These function values can be used to calculate the integral if we treat the MCMC samples in parameter space as an irregular grid. A straightforward way to do this is to calculate the Voronoi tessellation for the samples in the parameter space, an example of which is given in Figure 5.2. The samples are then the centres of the

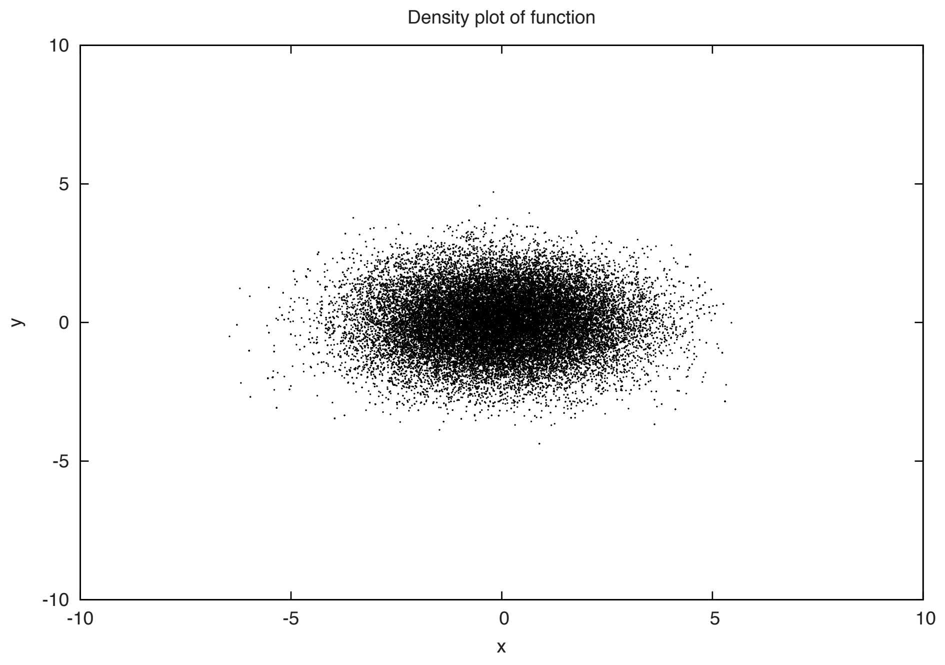

Figure 5.1: A scatter plot of of 40000 samples, drawn using a Metropolis algorithm from the function $f(x, y) = \exp(-ax^2 - by^2)$, with $a = 1/5$ and $b = 2/5$.

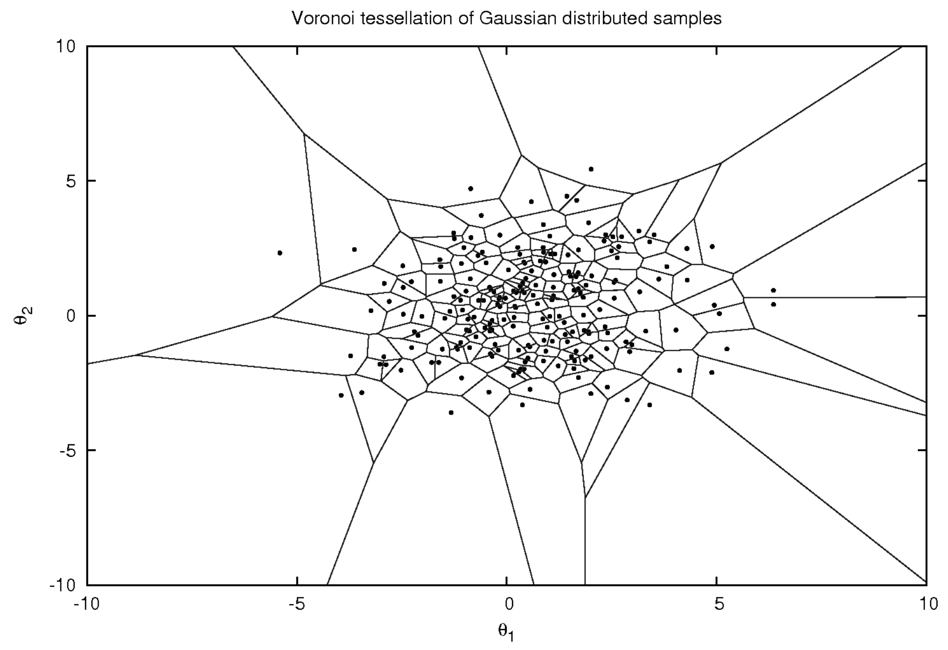

Figure 5.2: An example of a Voronoi tessellation. Here we have taken 200 samples from a 2-dimensional Gaussian distribution as the centres of the Voronoi diagram.

Voronoi cells, and the integral can be calculated as follows:

$$z \approx \sum_i f\left(\vec{\Theta}_i\right) O_i, \tag{5.11}$$

where O_i is the surface of the i^{th} Voronoi cell, and we only sum over closed cells (with finite surface). This procedure converges to the correct value for large number of samples N, and for all tessellations, not just when we use Voronoi cells. In practice, although Voronoi tessellations can be computed for any dimensionality, this becomes computationally prohibitive in practice for problems with high dimensionality (Edelsbrunner & Shah, 1996). This procedure does illustrate that all information needed to evaluate the integral z is present in the MCMC chain.

We now proceed to show that we can link the estimator of Equation (5.8) to the voronoi tessellation. First, we consider the HPD region $\mathcal{V}$ of $f(\vec{\Theta})$, and write the volume V as a sum of the volumes of the voronoi cells:

$$V = \sum_{i:\vec{\Theta}\in\mathcal{V}} O_i. \tag{5.12}$$

These cells comprise an inhomogeneous Poisson Voronoi Tessellation (PVT). Analytical expressions about the statistics of inhomogeneous PVTs are quite difficult to establish; most results are based on large simulations. Analytically it is known that $\langle O_i \rangle \to 1/$ density of points for $N \to \infty$, for cells away from the boundary (Barr, 2008). For finite N this expectation is biased. We ignore the bias, and introduce a proportionality constant α_N as follows:

$$\langle O_i \rangle = \frac{\alpha_N}{f(\vec{\Theta}_i)}. \tag{5.13}$$

By only considering the HPD region $\mathcal{V}$, we do not have to deal with cells with infinite volume. We can estimate α_N analogous to Equation (5.11), but now with the expectation of O_i:

$$\sum_{i:\vec{\Theta}\in\mathcal{V}} f(\vec{\Theta})\langle O_i \rangle = \alpha_N \sum_{i:\vec{\Theta}\in\mathcal{V}} 1 = \alpha_N N_{\mathcal{V}} \approx zw, \tag{5.14}$$

where we have used notation as in Section 5.3.2. Since the expectation of a sum is the sum of the expectation, and using Equations (5.14 & 5.13): we also have:

$$V = \sum \langle O_i \rangle \approx z \frac{w}{N_{\mathcal{V}}} \sum_{i:\vec{\Theta}\in\mathcal{V}} \frac{1}{f(\vec{\Theta})}. \tag{5.15}$$

This expression directly leads us to the estimator of Equation (5.8) that we have derived in the previous section.

5.3.4 A practical algorithm

In this section we construct a simple practical algorithm for numerically calculating integrals using regular MCMC methods. As stated in Section 5.3.3, we need to define a HPD region of the parameter space that is sufficiently populated by MCMC samples. In the case of multimodal posteriors, we should take samples that 'belong' to one specific mode. Clustering algorithms like X-means (Pelleg & Moore, 2000), and G-means (Hamerly & Elkan, 2003) can be used for this.

Assuming that we have found M samples that belong to a specific mode, we can then define a HPD region as follows. Assume that the M samples are sorted according to their function value $f(\vec{\Theta}_i)$, with $f(\vec{\Theta}_0)$ the highest value. We then first approximate the maximum of $f(\vec{\Theta})$ as:

$$\vec{\mu} = \frac{1}{k}\sum_{i=1}^{k}\vec{\Theta}_i, \tag{5.16}$$

where $k = aM$ is a small fraction of M, dependent on the shape of the mode. We use $a = 1/20$ in several cases. We now define our HPD region $\mathcal{V}$ as an ellipsoid with centre $\vec{\mu}$, and covariance matrix C, defined by:

$$C = \frac{1}{n}\sum_{i=1}^{n}\left(\vec{\Theta}_i - \vec{\mu}\right)\left(\vec{\Theta}_i - \vec{\mu}\right)^T, \tag{5.17}$$

where $n = bM$ is a fraction of M, again dependent on the shape of the mode. We use $b = 1/5$ in several cases. All samples within this ellipsoid of size $\sqrt{r^2 \det C}$ satisfy:

$$\left(\vec{\Theta} - \vec{\mu}\right)^T C^{-1}\left(\vec{\Theta} - \vec{\mu}\right) \leq r^2. \tag{5.18}$$

We adjust the parameter r^2 such that $l = cM$ samples satisfy this relation. It is crucial that we choose l to be as large as possible, while still satisfying the requirement that the entire ellipsoid is sufficiently populated with samples. If l is too small, we will not achieve very high accuracy for our integral z, but if l is too large, we have underpopulated parts of the ellipsoid which results in the wrong outcome. We use $c = 1/3$ in several cases.

We now have all the ingredients needed to calculate the integral z, as the Volume of our k-dimensional HPD region is given by:

$$V = \frac{r^k \pi^{\frac{k}{2}}}{\Gamma\left(1+\frac{k}{2}\right)} \sqrt{\det C}. \tag{5.19}$$

This, combined with Equation (5.8) allows us to evaluate the integral.

We would like to note that this prescription is merely one of many that could be used. In this case we have defined our HPD region as an ellipsoid located at a maximum of our integrand, but any other HPD region will suffice.

5.3.5 Error estimates

An ideal MCMC algorithm generates samples according to a PDF proportional to $f(\vec{\Theta})$. Consider the number of samples inside the HPD region $\mathcal{V}$. This number follows a Poissonian distribution with mean and variance equal to cM. Using that we can obtain a rough estimate for the error in the integral:

$$\Delta z = \frac{1}{\sqrt{cM}} z. \tag{5.20}$$

We note that this Equation (5.20) is too simplistic for the truncated harmonic mean estimator we introduce in chapter, but it can be used as an order of magnitude guide to how well a particular simulation will do. Even if the estimator follows the central limit theorem, which we do not show here, we show in Section 5.5 that the error estimate should depend on the samples of the MCMC algorithm that have been used. Many MCMC algorithms produce correlated samples, and in that case the error estimate of the ML estimator should also increase (Roberts et al., 1997).

Having efficiency as one of our goals, we would like to produce reliable errorbars on our numerical integral using the correlated MCMC samples. We propose to use a batch means method (Efron, 1979). In the case of several chains, one can use the spread in the estimates based on different chains to estimate the error of the integral. In the case of a single chain, we propose to divide the chain in 10 succeeding parts, and use the spread of the integral in those 10 parts to estimate the error of the numerical integral. In Section 5.5.4 we test the error estimates discussed here extensively.

Another point noting is that we expect the truncated harmonic mean estimator of Equation (5.8) to be slightly biased. Because the denominator of Equation (5.8) is an unbiased estimator, this does not hold for the ML estimator itself. We investigate the bias of the ML estimator in Section 5.5.4.

5.4 Comparison to other methods

Although the algorithm developed in this chapter is quite generally applicable, in practice there are caveats for each integration problem that one needs to be aware of in order to choose the right integration algorithm. In this section we discuss the strengths and the weaknesses of the method developed in this chapter in combination with the Metropolis Hastings MCMC algorithm, and we compare it to two other methods widely used in astronomy: Nested sampling, and Thermodynamic Integration based on Parallel Tempering (TI-PT).

For all algorithms, the main criteria that we cover are efficiency, accuracy and robustness.

5.4.1 Metropolis Hastings

The main advantage of the Metropolis Hastings (MH) algorithm in relation to ML evaluation is that of parameter space reduction; the full parameter space over which we would like to evaluate the integral is too large, which is why we cannot use a regular grid. The MH algorithm will mostly sample in the HPD regions, the regions of parameter space where the function value $f(\vec{\Theta})$ is high enough to significantly contribute to the integral. In general this can be done by drawing samples from a distribution $P(\vec{\Theta})$ that resembles the function $f(\vec{\Theta})$, with the optimal choice being $P(\vec{\Theta}) = f(\vec{\Theta})$. MH is specifically designed to satisfy this optimal relation, making the MH algorithm optimally efficient from a theoretical point of view (Roberts et al., 1997). In Figure 5.1 we show an example of samples generated with a MH algorithm, released on a 2-dimensional Gaussian function.

The drawback of MH is that one can never be sure that all important parts of the parameter space have been covered. Some functions have many distinct peaks, the so-called modes of the function. The MH algorithm often has difficulty to make the chain move from one mode to another. If the function $f(\vec{\Theta})$ is highly multimodal, we must make sure that the sample density ratio between the modes is right and that we have reached all the important modes in the entire parameter space.

An additional practical challenge with the method developed in this chapter is to make sure that we have constructed a HPD region that is sufficiently populated with samples. In the case of a not very peaked, or highly degenerate function this requires special care.

5.4.2 Nested sampling

Nested sampling is a Monte Carlo technique aimed at accurate evaluation of numerical integrals, while staying efficient (Skilling, 2004). The algorithm is especially well suited for problems that have huge parameter spaces, and very complicated multi modal distributions. Nested sampling starts with sampling from the original parameter space; in the case of ML calculation this is equivalent to sampling from the prior distribution of the parameters. The density with which the parameter space is sampled is adjustable in the form of an number n_L of so-called live points; the number of points evenly distributed among the part of the parameter space we are exploring. At the start of the algorithm, the live points are evenly distributed over the entire parameter space. Then the live points are replaced one-by-one under the restriction that the newly sampled live points are higher than the lowest one we have not replaced yet. Effectively this is the same as shrinking the parameter space by a fixed factor every time we replace a new live point, ultimately sampling only the part of the function close to the maximum.

The main advantage of nested sampling is that one generates samples from the original parameter space, thereby completely eliminating the main deficiency of the MH algorithm: all modes of the function $f(\vec{\Theta})$ are sampled from, provided that we choose n_L high enough, i.e. that we have a high enough sampling density in the parameter space. And due to the shrinking of the parameter space by a fixed factor with the draw of each new live point, the size of the parameter space decreases exponentially. This way all of the parameter space is explored in a relatively manageable way, making nested sampling quite efficient in searching the parameter space for HPD regions.

The main disadvantage of nested sampling is that the samples are not drawn from a distribution that closely resembles the function $f(\vec{\Theta})$ we want to integrate. This method will therefore never reach the efficiency that the MH algorithm offers. In Figure 5.3 we show an example of samples generated with nested sampling, applied to a 2-dimensional Gaussian function.

5.4.3 Parallel tempering

Thermodynamic Integration based on Parallel Tempering (TI-PT) is an algorithm spawned from the desire to solve the problems of MH and similar MCMC methods. TI-PT algorithms possess better mixing properties, allowing the chain to “escape” local extrema, and allow one to calculate the complete integral, or in our case the ML (Earl & Deem, 2005). Let us briefly review TI-PT in this section, without discussing it in too much detail.

The main idea of TI-PT is that of parameter space exploration by adding an imaginary inverse temperature β to the system, changing the integrand of our integral to:

$$f_{\beta}\left(\vec{\Theta}\right) = \left(f(\vec{\Theta})\right)^{\beta} . \tag{5.21}$$

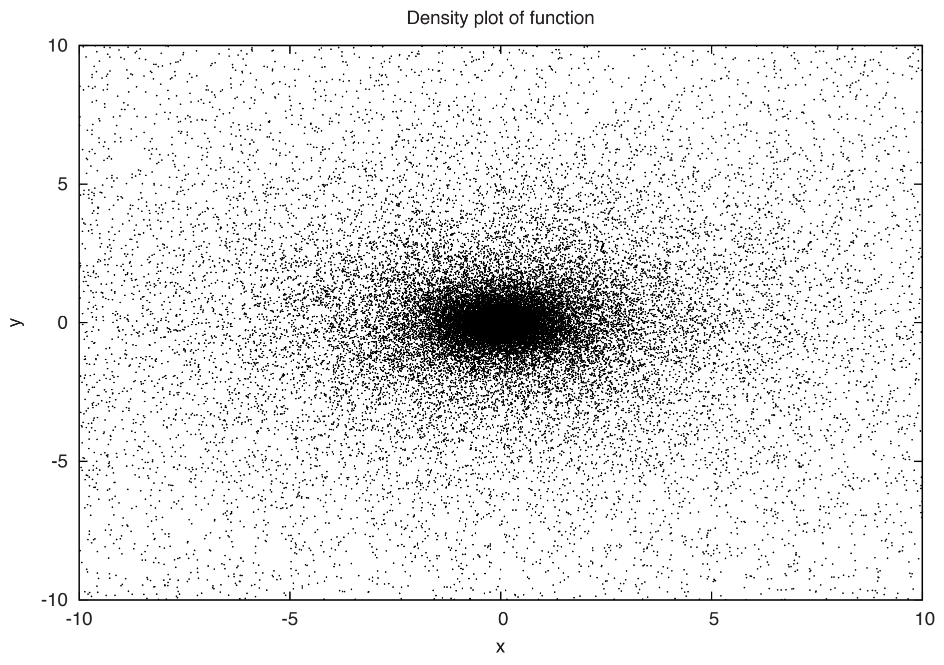

Figure 5.3: A scatter plot of 40000 samples, drawn using a nested sampling algorithm with 5000 live points from the function $f(x, y) = \exp(-ax^2 - by^2)$, with $a = 1/5$ and $b = 2/5$.

Then many MCMC chains are released in the parameter space, each with a different temperature $\beta \in [0, 1]$. A clever swapping system is employed, allowing the chain with $\beta = 1$ - the "true" chain - to swap parameters with chains of higher temperature every now and then provided that such a high-temperature chain was able to reach a point in parameter space with high $f(\vec{\Theta})$. This trick allows the coldest system, the "true" chain with $\beta = 1$, to escape local extrema.

The integral over $f(\vec{\Theta})$ is calculated by using all chains, not just the one with $\beta = 1$, as follows. We first define a partition function:

$$Z(\beta) = \int \mathrm{d}\vec{\Theta} f_\beta\left(\vec{\Theta}\right), \tag{5.22}$$

which has a logarithmic derivative of:

$$\frac{\mathrm{d}}{\mathrm{d}\beta} \log\left(Z(\beta)\right) = \frac{1}{Z(\beta)} \frac{\mathrm{d}}{\mathrm{d}\beta} Z(\beta) = \left\langle \log\left(f(\vec{\Theta})\right)\right\rangle_\beta, \tag{5.23}$$

where $\langle . \rangle_\beta$ is the expectation value of a quantity over a distribution proportional to $f_\beta(\vec{\Theta})$. Since we know that our desired integral can be expressed as $z = Z(1)$, Equation (5.23) is all we need to calculate it:

$$\log\left(Z(1)\right) = \log\left(Z(0)\right) + \int_0^1 \mathrm{d}\beta \left\langle \log\left(f(\vec{\Theta})\right)\right\rangle_\beta. \tag{5.24}$$

The observant reader will have noted that we have neglected to mention the size $Z(0)$ of the parameter space that we explore with our chains. The high temperature chain with $\beta = 0$ is unbounded by the function $f(\vec{\Theta})$, and therefore will transverse the entire parameter space. We should make sure that we limit the size of the parameter space as much as possible, without missing any peaks of $f(\vec{\Theta})$.

The main advantages of TI-PT are that it explores the entire parameter space, even in the presence of strong local peaks, and that the ML can be calculated. These advantages are accompanied with the large computational costs of the extra chains with $\beta \neq 1$, which has resulted in the need for alternative methods like nested sampling. As MultiNest, a specific nested sampling algorithm, is supposed to outperform TI-PT in virtually all cases (Feroz et al., 2009), we compare our method to MultiNest only in Section 5.5.

5.5 Applications and tests

In this section we consider several toy models, and we apply several integration algorithms to these toy models as to compare the algorithms. In this whole section we have two ways to use the MH algorithm. To produce correlated samples, we use a MH algorithm where we use the entire chain. To produce uncorrelated samples, we use the same algorithm, but we only store one in every j samples produced by the algorithm. We choose j high enough that there is negligible correlation between 2 succeeding used samples; in the case of a 4-dimensional Gaussian as in Section 5.5.1 we use $j = 100$.

5.5.1 Toy model 1: a high-dimensional Gaussian

We first consider a problem that is a typical example of what MH algorithm are efficient in: a highly peaked, high-dimensional function. The curse of dimensionality prohibits any direct numerical integration scheme on a fine grid, but analytical integration is possible. Consider the multiplication of N Gaussian functions,

$$f_1(\vec{x}) = \prod_{i=1}^{N} \sqrt{\frac{a_i}{2\pi}} \exp\left(-\frac{1}{2} a_i x_i^2\right), \tag{5.25}$$

with a_i the width of the Gaussian in the i^{th} direction. Now let us perform a volume preserving coordinate transformation using a random orthogonal matrix R, generated with a QR decomposition of a random matrix, as follows: $\vec{\Theta} = R\vec{x}$. If we introduce a matrix $A^{-1}_{ii} = a_i = 1 + i$ with $A^{-1}_{ij} = 0$ for $i \neq j$, we then have a highly degenerate high-dimensional Gaussian function:

$$f_1\left(\vec{\Theta}\right) = \frac{1}{(2\pi)^{N/2}\sqrt{\det C}} \exp\left(-\frac{1}{2}\vec{\Theta}^T C^{-1} \vec{\Theta}\right), \tag{5.26}$$

where we have introduced the covariance matrix $C = RAR^T$.

We now apply in turn the MultiNest nested sampling algorithm and the algorithm developed in this chapter combined with MH to the multi-dimensional Gaussian for various number of dimensions.

For the MultiNest algorithm, we use a number of live points $L = 50N$ with N the number of dimensions, and we have set the sampling efficiency to $e = 0.3$ and the ML tolerance to $t = 0.1$ as advocated in (Feroz et al., 2009).

We set up the Metropolis-Hastings algorithm to have an acceptance ratio equal to the optimal value of 23.4% (Roberts et al., 1997). The parameters of the algorithm advocated in Section 5.3.4 have been set to: $a = 1/20, b = 1/5, c = 1/3$. We have used the number of samples N used by the MultiNest algorithm as the number of samples in our MCMC chain. However, we realise that the MultiNest algorithm might be improved in the future, as the algorithm is still under construction. The lowest efficiency (used samples / drawn samples) we encountered for log (z) for different algorithms

n	# N	MultiNest	Unc. MCMC	Cor. MCMC
2	2902	-0.17 ± 0.18	-0.018 ± 0.025	0.03 ± 0.025
4	7359	0.20 ± 0.17	0.007 ± 0.024	-0.01 ± 0.03
8	24540	0.17 ± 0.17	-0.01 ± 0.01	0.02 ± 0.03
16	10^5	0.05 ± 0.18	0.001 ± 0.006	0.004 ± 0.03
32	10^6	0.43 ± 0.17	0.004 ± 0.004	-0.015 ± 0.010
64	4.10^6	1.07 ± 0.77	-0.0004 ± 0.0007	-0.02 ± 0.016

Table 5.1: The log-integral values of the function f_1 of Equation (5.26). The analytically integrated value is $\log z = 0$ for all values of N.

Jump 1&2	Jump 3&4	Jump 5&6	Jump 7&8
$\Theta_1 \pm 4\pi$	$\Theta_1 \pm 2\pi$	$\Theta_1 \pm 2\pi$	$\Theta_1 \pm 0$
$\Theta_2 \pm 0$	$\Theta_2 \pm 2\pi$	$\Theta_2 \mp 2\pi$	$\Theta_2 \pm 4\pi$

Table 5.2: The possible jumps in the eggbox MCMC toy-problem.

this toy-problem was $e = 0.08$; we therefore estimate that the error-bars could be decreased with a factor of maximally $\sqrt{1/0.08} \approx 3.5$. The results of this toy-model are shown in table 5.1.

5.5.2 Toy model 2: egg-box function

Just as in FHB09, we now consider a highly multimodal two-dimensional problem for which the function resembles an egg-box. The function is defined as:

$$f_2\left(\vec{\Theta}\right) = \exp\left[2 + \cos\left(\Theta_1\right)\cos\left(\Theta_2\right)\right]^5, \tag{5.27}$$

where we set the domain of the function equal to $[0, 10\pi]$ for both parameters. The shape of this function is shown in Figure 5.4. This is a typical problem where difficulties arise for traditional MCMC algorithms. Many solutions have been proposed for situations like this (Newman & Barkema, 1999), but in practice one needs to have additional information about the problem for any of those solutions to be reliable. For the sake of clarity of this chapter, we do not concern us with the practical implementation of the MCMC algorithm. We assume that a suitable trick can be found for the problem at hand so that the algorithm proposed in this chapter can be used. For the eggbox toy-model we will use a jump-technique. At each iteration of the MCMC algorithm, there is a small probability, 1% in this case, that the chain will jump to a neighbouring mode. The available jumps are shown in table 5.2.

The MultiNest algorithm is ideally suited for this problem, as this is a low-

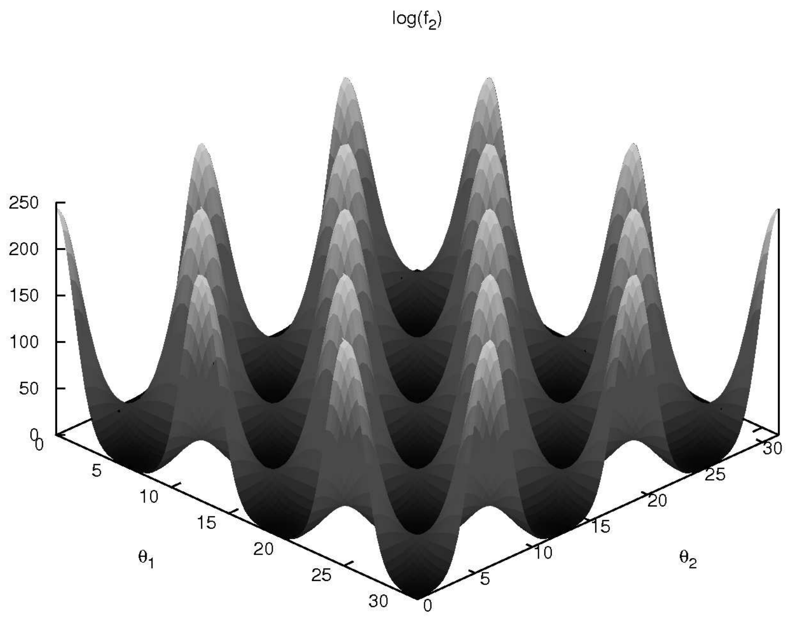

Figure 5.4: Toy-model 2: the eggbox of Equation (5.27)

log (z) for different algorithms			
# N	MultiNest	Unc. MCMC	Cor. MCMC
60210	240.19 ± 0.05	240.19 ± 0.027	240.23 ± 0.05

Table 5.3: The log-integral values of a single mode of the eggbox function of Equation (5.27) The fine-grid integrated value is $\log z = 240.224$

dimensional multimodal function. With enough live samples all modes should be easily discovered, and the peaks are symmetric and well-behaved. We run Multi-Nest on this problem with the same parameters as in FHB09: We use $L = 2000$ live points, efficiency $e = 0.3$, and tolerance $t = 0.1$.

Table 5.3 shows the results of this analysis[1]. For evaluating the integral with the MCMC chain, we have taken a total of $N = 60210$ samples as was done with Multi-Nest, but we have used only the samples of one single peak in Equation (5.8). The number of samples in a single peak is $2/25$ of the total number of samples, leading to loss of accuracy. Though more sophisticated methods can be constructed by, say, averaging the ML estimators for several HPD regions for all individual peaks, we show here that we only need to find one single HPD region that is sufficiently populated to calculate a reliable value for the integral.

5.5.3 Application in Pulsar Timing

In pulsar timing data analysis, one often encounters datasets of which the exact statistics are not well known. Bayesian model selection would provide the ideal tool to obtain information about the power spectra present in the data. van Haasteren et al. (2009, see also chapter 2) give a full description of the Bayesian data analysis for pulsar timing arrays, but their work lacks a method to calculate the ML from the MCMC samples. In this section, we use a pulsar timing mock dataset to show that the method developed in this chapter is well-suited for ML calculation in pulsar timing problems. We also use MultiNest to analyse this dataset, and we compare the two results.

For the purposes of this chapter, the description of the pulsar timing posterior distribution is kept brief; full details can be found in chapter 2. The data of pulsar timing experiments consists of the arrival times of pulsar pulses (TOAs), which arrive at the earth at highly predictable moments in time (Hobbs et al., 2006). The deviations from the theoretically predicted values of these TOAs are called the timing-residuals (TRs). These TRs are the data we concern ourselves with in this example.

[1]The true value presented here is different from the one noted in FHB09. We have incorporated hypercube transformation into our results.

Consider $n = 100$ TRs, denoted as $\vec{\delta t}$, observed with intervals between succeeding observations of 5 weeks. Assume that we are observing a stable and precise millisecond pulsar with timing accuracy about $\sigma = 100$ns (the error bars on the TRs). Usually σ not precisely known, since pulsar timers generally assume that their estimate of the error bar is slightly off. Several datasets of millisecond pulsars also seem to contain correlated low frequency noise (Verbiest et al., 2009). We therefore also allow for some correlated timing-noise in the data, with a power-spectrum given by:

$$S(f) = r^2 \gamma \exp(-\gamma f), \tag{5.28}$$

where f is the frequency, r is the amplitude of the correlated timing-noise in ns, and γ is the typical size of the structures that appear in the data due to this correlated timing-noise. Following chapter 2, we can now write the likelihood function for the TRs as a multi-dimensional Gaussian:

$$P\left(\vec{\delta t} \mid \sigma, r, \gamma\right) = \frac{1}{\sqrt{(2\pi)^n \det C}} \exp\left(-\frac{1}{2}\vec{\delta t}^T C^{-1} \vec{\delta t}\right), \tag{5.29}$$

where C is an $(n \times x)$ matrix, with elements defined as:

$$C_{ij} = \sigma^2 \delta_{ij} + r^2 \frac{\gamma^2}{\gamma^2 + \tau_{ij}^2}, \tag{5.30}$$

with δ_{ij} the Kronecker delta, and τ_{ij} is the time difference between observation i and observation j.

Simulating a mock dataset from such a Gaussian distribution is quite straightforward; for details see chapter 2. We now analyse a mock dataset, shown in Figure 5.5, generated with parameters: $\sigma = 100$ns, $r = 100$ns, and $\gamma = 2$yr. We assume uniform prior distributions for all parameters: $\sigma \in [0, 1000]$ns, $r \in [0, 1000]$ns, and $\gamma \in [0, 10]$yr. The posterior is then sampled using both MultiNest and a Metropolis-Hastings algorithm, resulting in marginalised posterior distributions as shown in Figure 5.6 &5.7. ML values are in good agreement between the two methods:

$$\begin{aligned} z_{\text{MCMC}} &= \exp(1523.12 \pm 0.17) \\ z_{\text{MultiNest}} &= \exp(1522.93 \pm 0.15). \end{aligned} \tag{5.31}$$

For both methods, the same number of samples has been used: $N = 9617$.

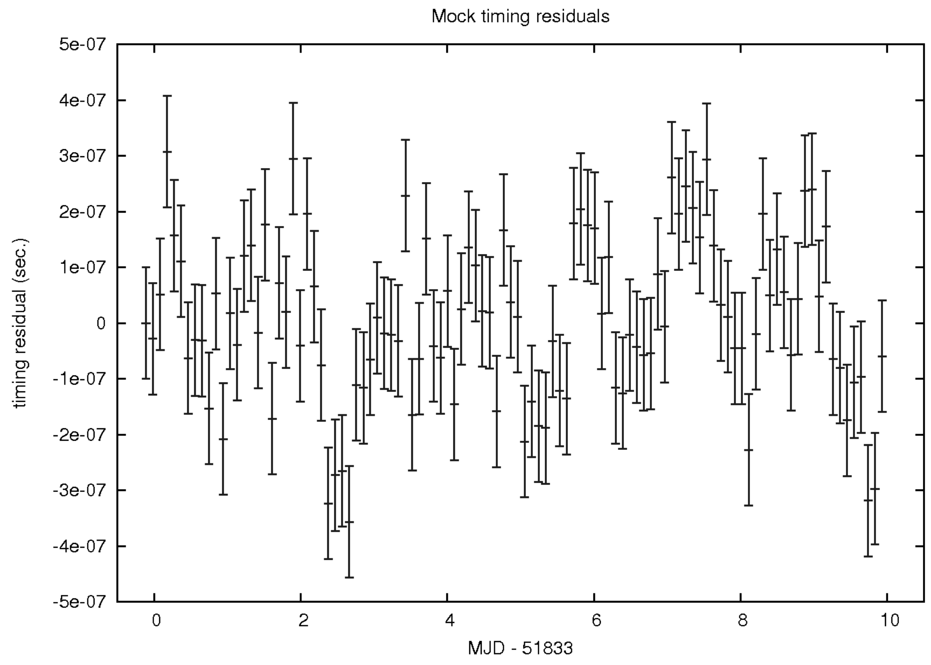

Figure 5.5: The mock timing-residuals we have analysed with both MultiNest and the method developed in this chapter

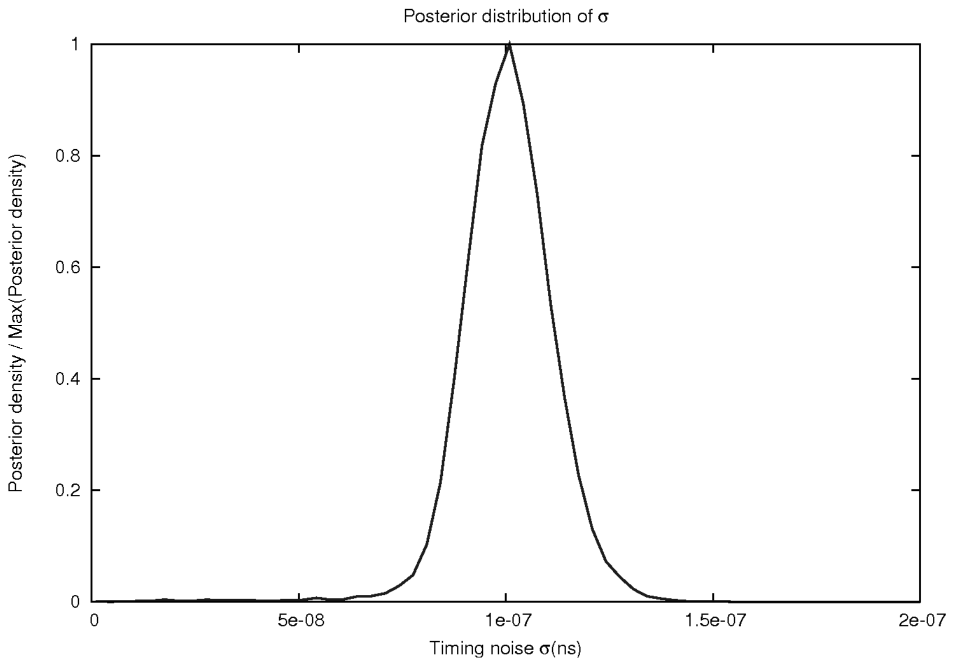

Figure 5.6: The marginalised posterior of the a parameter, sampled using a Metropolis-Hastings MCMC method.

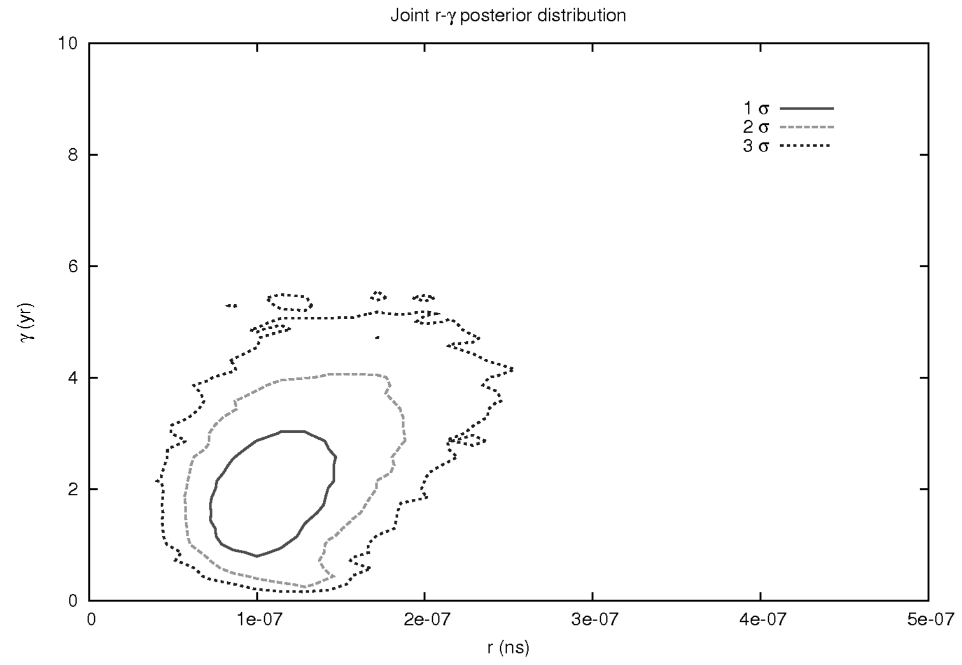

Figure 5.7: The marginalised posterior of the r and γ parameters, sampled using a Metropolis-Hastings MCMC method.

5.5.4 Assessing bias

We now test the accuracy and the bias of the algorithm by running it many times on the same toy-problem, and then considering the statistics of the ensemble. We found the 16-dimensional Gaussian of Section 5.5.1 to be an illustrative example. Just as in table 5.1, we take $N = 10^5$ and $c = 0.3$, and then we run $n = 10^4$ Metropolis-Hastings chains on this toy-problem. For the i^{th} chain we then calculate the integral z_i and batch means error estimate σ^{BM}. We have presented the results of this analysis as a histogram of z_i values in Figure 5.8. Several useful quantities that characterise the ensemble are:

$$
\begin{aligned}
\bar{z} &= \frac{1}{n}\sum_{i=1}^{n} z_i = 0.980 \\
\bar{\sigma} &= \sqrt{\frac{1}{n}\sum_{i=1}^{n}(z_i - \bar{z})^2} = 0.028 \\
\bar{\sigma}^{\text{BM}} &= \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\sigma_i^{\text{BM}}\right)^2} = 0.027,
\end{aligned}
\tag{5.32}
$$

where $\bar{z}$ is the integral average, $\bar{\sigma}$ is the rms of the integral values, and $\bar{\sigma}^{\text{BM}}$ is the rms value of the batch means errors.

Figure 5.8 shows that the batch means error estimate is quite correct, since $\bar{\sigma} \approx \bar{\sigma}^{\text{BM}}$. However, though smaller than $\bar{\sigma}$, there is a significant deviation in the value of of $\bar{z}$ compared to the true value $z = 1$. This shows that the estimator indeed has a small bias, as noted in Section 5.3.5.

In order to investigate the behaviour of the bias of $\bar{z}$, we perform 3 additional tests. In all 3 cases, we construct a new ensemble of MCMC chains identical to the ensemble above, except for one parameter. The differences with the above mentioned ensemble are:

1. Instead of a correlated MCMC chain, we use an MCMC chain of uncorrelated samples, produced by performing a regular MCMC but only storing every 100th sample.
2. $N = 10^7$, instead of $N = 10^5$. This results in much more samples in the HPD region.
3. $c = 0.7$, instead of $c = 0.3$, which also results in more samples in the HPD region.

We note that these results are merely illustrative of the behaviour of the ML estimator, since the bias of the estimator is problem dependent.

We present the results of this analysis as the values of Equation (5.32) in table

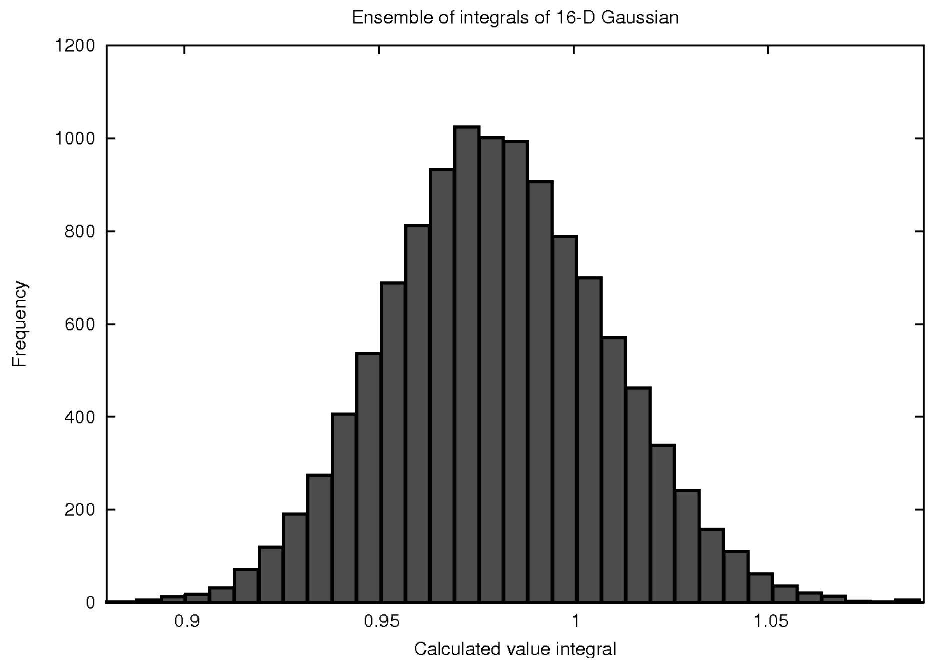

Figure 5.8: Histogram of the frequency of calculated integral using the method developed in this chapter. We have taken the 16-dimensional Gaussian of Equation (5.26) as integrand. Here we have analysed it with $N = 10^5$, and $c = 0.3$. This histogram has mean and standard deviation: $\bar{z} = 0.980$, $\bar{\sigma} = 0.028$. The rms of the batch means error of the ensemble was $\bar{\sigma}^{\mathrm{BM}} = 0.027$

Ensemble statistics for different parameters

Cor./Unc.	N	c	$\bar{z}$	$\bar{\sigma}$	$\bar{\sigma}^{\mathrm{BM}}$
Cor.	10^5	0.3	0.980	0.028	0.027
Unc.	10^5	0.3	1.000	0.006	0.006
Cor.	10^7	0.3	1.000	0.003	0.003
Cor.	10^5	0.7	0.994	0.020	0.034

Table 5.4: The statistics of Equation (5.32) for various ensembles of $n = 10^4$ MCMC runs on a 16-dimensional Gaussian. The first chain has the same parameters as used in section 5.5.1. The other chains differ in either number of samples per chain N, the size of the HPD region c, or whether or not the samples in a chain are correlated. Note that the error in the uncorrelated chain equals the limit of $\sigma = \sqrt{1/cN} = 0.006$.

5.4. All adjustments seem to reduce the bias of the estimator. Several notes are in order:
1. The only reason we can increase c is because we know exactly what the integrand looks like. In practical applications this is probably not an option. Also note that batch means error increases, indicating that estimate of the integral is less stable.
2. The fact that the uncorrelated chain performs as well as described by Equation (5.20) shows that the algorithm suffers significantly under the use of correlated MCMC chains.
3. Increasing the number of samples in a chain makes the calculated integral more reliable, as the HPD region is more densely populated. Note that this large chain is build up of small chains that would yield a more biased value.

5.6 Discussion and conclusions

We develop and test a new algorithm that uses MCMC methods to accurately evaluate numerical integrals that typically arise when evaluating the ML. The new method can be applied to MCMC chains that have already been run in the past so that no new samples have to be drawn from the integrand, provided that we can define a high probability density region of the integrand well enough based on the MCMC samples. We test the new algorithm on several toy-problems, and we compare the results to other methods: nested sampling and thermodynamic integration based on parallel tempering. We conclude that the new algorithm could be of great value for high-dimensional, peaked problems. When applicable, the new algorithm can outperform other algorithms, provided that the MCMC has been properly executed. This new algorithm is therefore expected to be useful in astrophysics, cosmology and particle physics.

We have demonstrated that the new algorithm suffers under the use of correlated MCMC chains, produced by using the entire chain of a particular MCMC method like Metropolis-Hastings. If the new algorithm is used in combination with an uncorrelated chain, the accuracy of the numerical integral can reach the value: $\sigma = z/\sqrt{N}$, with σ the uncertainty, z the value of the integral, and N the number of MCMC samples in the high probablility density region. Using correlated MCMC samples can significantly increase the integral uncertainty, and longer MCMC chains are needed for the integral to converge. We have also shown that the new estimator is slightly biased, where the bias is problem dependent. Additional tests to assess convergence and bias are required.