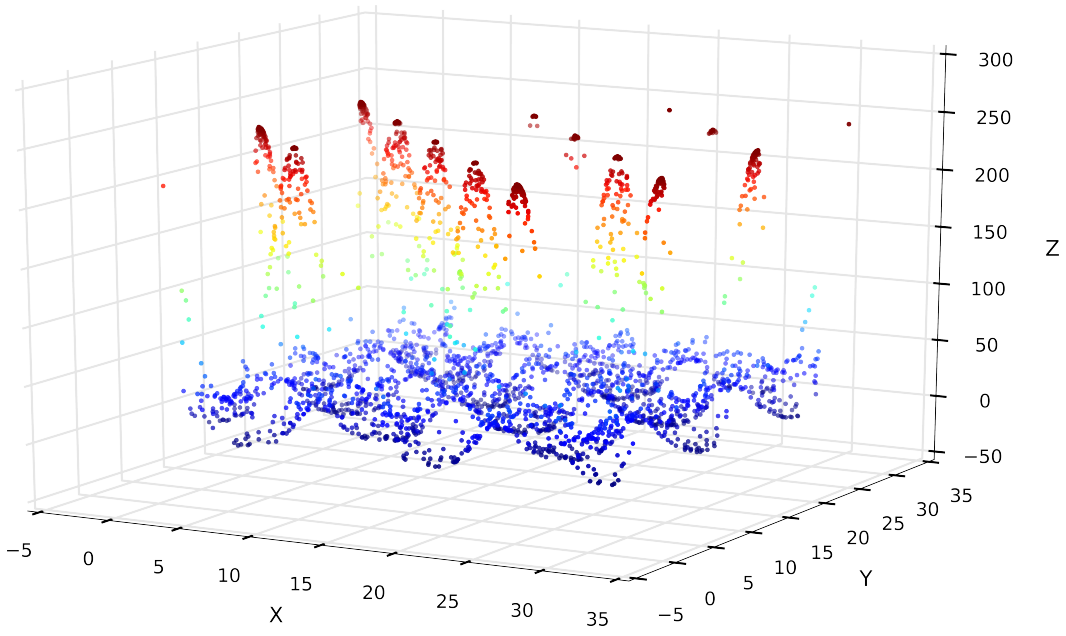


Scalability of Sampling in MultiNest

Maicon Hieronymus

January 26, 2016

Bachelor Thesis in Computer Science

Scalability of Sampling in MultiNest

Maicon Hieronymus

1st Reviewer Prof. Dr. Bertil Schmidt
Department of Computer Science
Johannes-Gutenberg University Mainz

2nd Reviewer Prof. Dr. Elmar Schömer
Department of Computer Science
Johannes-Gutenberg University Mainz

Supervisors Prof. Dr. Bertil Schmidt and Dr. Christian Hundt

January 26, 2016

Maicon Hieronymus
Scalability of Sampling in MultiNest
Bachelor Thesis in Computer Science, January 26, 2016
Reviewers: Prof. Dr. Bertil Schmidt and Prof. Dr. Elmar Schömer
Supervisors: Prof. Dr. Bertil Schmidt and Dr. Christian Hundt

Johannes-Gutenberg University Mainz
High Performance Computing
Department of Computer Science
Staudingerweg 9
D-55128 and Mainz

Abstract

MultiNest is a multimodal nested sampling algorithm from F. Feroz, M. P. Hobson and M. Bridges, which calculates the evidence in Bayesian analysis with a certain error estimate. For that one generates posterior samples from distributions which might have multiple modes and pronounced curves in high dimensions. MultiNest is used in various astrophysical inference problems, since it delivers in this particular context a better performance than traditional Markov Chain Monte Carlo (MCMC) methods. In this thesis we introduce three parallelization variants, in order to enhance the speed or the accuracy, and discuss these on the basis of three toy models. These approaches react especially to the concurrent evaluation of the likelihood because this evaluation demands the most running time in praxis.

Zusammenfassung

MultiNest ist ein multimodales Nested-Sampling Verfahren von F. Feroz, M. P. Hobson und M. Bridges, das die Evidenz in Bayes'schen Analysen mit einer gewissen Fehlertoleranz berechnet. Dafür werden Posterior-Samples aus Verteilungen generiert, die mehrere Moden und ausgeprägte Kurven in hohen Dimensionen beinhalten können. MultiNest wird in verschiedenen astrophysischen Inferenzproblemen genutzt, da es eine für diese spezielle Aufgabenstellung bessere Performance als traditionelle Markov Chain Monte Carlo (MCMC) Methoden bietet. In dieser Arbeit präsentieren wir drei Varianten zur Parallelisierung von MultiNest, um die Geschwindigkeit oder die Genauigkeit zu verbessern, und diskutieren diese anhand von drei Toy-Models. Die Ansätze gehen besonders auf eine parallele Auswertung der Likelihood ein, da diese Auswertung in der Praxis die meiste Laufzeit beansprucht.

Acknowledgement

At first I would like to thank Prof. Dr. Schmidt. His lectures inspired me to learn more about High Performance Computing. In addition I would like to thank Prof. Dr. Böser for providing an interesting topic and giving me a fascinating insight in his work. It is great to see a field of application for my very first thesis. He also gave me access to the repository of IceCube in order to test my code and he provided me my very own work desk.
I would like to thank Thomas Erhardt who is currently making his PhD at IceCube with Prof. Dr. Böser for answering all questions I had and for helping me to understand the purpose of MultiNest.
Next I would like to thank my supervisor Dr. Christian Hundt, who helped me to get rid of my frustration during my work. His knowledge and ideas motivated me and I still could continue to work on this topic in order to implement and test some more.
At last I would like to thank André Müller who is currently making his PhD at the department of Computer Science for explaining me the benefits of traditional Markov Chain Monte Carlo methods.

Contents

1 Introduction

> *A billion neutrinos go swimming in heavy water: one gets wet.*
>
> — **Michael Kamakana**

MultiNest is a Bayesian inference tool which became more popular in a wide range of astrophysical inference problems due to its superior accuracy and speed compared to traditional Markov Chain Monte Carlo (MCMC) methods in some cases (see e.g. [23], [7], [32], [13], [10], [4], [16], [20] and [6]). Nonetheless, some problems need a lot of time even with MultiNest for example analyzing the energy loss of muons due to Cherenkov-radiation in order to reconstruct the beta-decay of neutrinos as done by the IceCube colaboration which is the inspiration for this thesis (see [1] for an approach to evaluate the likelihood on GPUs and [17] for a good explanation of the reconstruction). While profiling the code used in IceCube we noticed that about 99.634% of the runtime is spent in calculating the likelihood for each sampled point. This is a result of the highly noninteractive nature of neutrinos. As a consequence we focused on parallelizing the sampling within MultiNest in order to gain the highest speedup.
In Chapter 2 we are going to explain why Bayesian statistics is used and how MultiNest works. After that chapter we outline the different approaches we used and in Chapter 4 and 5 some toy models and the results are presented and discussed. At last we give a conclusion and some ideas on how to improve MultiNest further.

Bayesian Analysis & MultiNest

2

> *Null-hypothesis tests are not completely stupid, but Bayesian statistics are better.*
>
> **— David Rindskopf**
> (1998)

2.1 Bayesian Analysis

In classical statistics one tests a hypothesis by comparing measured data with the expected outcome if the hypothesis is true. The better both match the more one can trust the hypothesis. This null-hypothesis test does not provide a probability if the hypothesis is true or not since a hypothesis is true or false despite the measured data [31]. This can be shown with Bayes' theorem that states

$$P(H|A) = \frac{P(A|H) \cdot P(H)}{P(D)} \tag{2.1}$$

where H is the event where the hypothesis is true, D is the event where the measured data falls within the range of the assumption and $P(H|A)$ is called the *posterior*. In this equation $P(A|H)$ is the *likelihood* that the measured data is within the range of the assumption if the null-hypothesis is true. This can be controlled by choosing $P(A|H) = 1 - \alpha$. It has been established as standard that $\alpha = 0.05$ is used for a significant conclusion, $\alpha = 0.01$ for a very significant conclusion and $\alpha = 0.001$ is used for a highly significant conclusion [3]. $P(D)$ is the *evidence* and describes the probability if measured data falls within range of our assumption independent of the null-hypothesis being true or not and $P(H)$ is the *prior* and represents the probability whether the null-hypothesis is true. One usually does not know if the null-hypothesis is true and even if the values are known beforehand, null-hypothesis tests do not take foreknowledge into account [31]. All in all a null-hypothesis test states, if a sample roughly fits a hypothesis then it should not be rejected. A wrong null-hypothesis might not get rejected with different samples but it still remains a wrong null-hypothesis.

In Bayesian analysis a test returns a probability whether a hypothesis is true or not. Rejecting or accepting the hypothesis is not part of the Bayesian analysis and belongs to decision theory. First we give some definitions before we state the benefits of

Bayesian analysis.
The posterior probability distribution $P(\Theta|\mathbf{D}, H) = P(\Theta)$ of the set of parameters Θ in a hypothesis H (also called model) represents the current knowledge about the model parameters with respect to the measured data D. $P(\Theta|H) = \pi(\Theta)$ is called the prior which contains the current knowledge about Θ. If nothing is known about the prior distribution Θ of the parameters, one takes a uniform distribution. $P(\mathbf{D}|\Theta, H) = \mathcal{L}(\Theta)$ is the likelihood to observe D given Θ and H. $P(\mathbf{D}|H) = \mathcal{Z}$ is the Bayesian evidence which can usually be ignored in parameter estimation since it is independent of the parameters Θ. In model selection however the evidence is needed to normalize the posterior over Θ via

$$\begin{aligned} \int \mathcal{L}(\Theta)\pi(\Theta)d^D\Theta = \int \mathrm{P}(D|\Theta, H) \cdot P(\Theta, H)d^D\Theta = \mathrm{P}(D|H) = \mathcal{Z} \\ \Rightarrow \mathcal{Z} = \int \mathcal{L}(\Theta)\pi(\Theta)d^D\Theta, \end{aligned} \tag{2.2}$$

where D is the dimensionality of the parameter space [8]. With these new definitions Bayes' theorem becomes

$$P(\Theta|\mathbf{D}, H) = \frac{P(\mathbf{D}|\Theta, H) \cdot P(\Theta|H)}{P(\mathbf{D}|H)}. \tag{2.3}$$

The evidence and the average of the likelihood over the prior are larger if more of its parameter space is likely and smaller for a model with large areas in its parameter space with low likelihood values even if the likelihood function is very highly peaked [8]. Therefore a compact parameter space will be preferred over a more complicated one if the latter one has big regions with low likelihood even if some regions are very highly peaked (usually called the modes of the likelihood region) implicitly implementing Occam's razor.
Comparing two models H_0 and H_1 can be done by comparing the posterior probabilities given the measured data set $\mathbf{D}$ by evaluating the Bayes factor (see [21])

$$\frac{\mathrm{P}(H_1|\mathbf{D})}{\mathrm{P}(H_0|\mathbf{D})} = \frac{\mathrm{P}(\mathbf{D}|H_1) \cdot \mathrm{P}(H_1)}{\mathrm{P}(\mathbf{D}|H_0) \cdot \mathrm{P}(H_0)} = \frac{\mathcal{Z}_1 \cdot \mathrm{P}(H_1)}{\mathcal{Z}_0 \cdot \mathrm{P}(H_0)}. \tag{2.4}$$

In this case $\mathrm{P}(H_1)/\mathrm{P}(H_0)$ is the a priori probability ratio, which can often be set to unity [8]. Evaluating the integral from Equation 2.2 is a very expensive task and is usually done via simulated annealing with thermodynamic integration which uses a Markov Chain Monte Carlo (MCMC) method. This needs a sufficient amount of samples in the prior space. One can obtain accuracies within 0.5 units in log-evidence but in cosmological applications it requires of order 10^6 samples per chain (with around 10 chains required to determine a sampling error) [8]. Skilling (see [30]) stated that thermodynamic integration needs the log-likelihood to be concave and has problems in navigating through convex or not differentiable log-likelihood regions. This is the reason MultiNest has been developed.

2.2 MultiNest

MultiNest is a Monte Carlo method for evaluating the Bayesian evidence and producing the posterior inferences as a by-product. In the paper *'Challenges of Profile Likelihood Evaluation in Multi-Dimensional SUSY Scans'* [9] the authors concluded that MultiNest can approximate the profile likelihood functions as well which makes MultiNest an expectation maximation algorithm. In this section we are going to explain nested sampling first and then extend it with ellipsoidal decomposition as explained in [8]. After that we explain the enhancements given by [12]. At last we present a short pseudocode for MultiNest as it is implemented in MultiNest v3.9.

Algorithm 1 evidence_evaluation()

$i \leftarrow 0$ ▷ Iteration counter at begin
$\mathcal{Z} \leftarrow 0$ ▷ Initialize the evidence
livepoints←draw_samples($\pi(\Theta)$, N) ▷ Draw N samples from full prior
▷ The prior volume is $X_0 = 1$ at the beginning
while prior volume has not been traversed entirely **do**
  $i \leftarrow i + 1$
  sort(livepoints, ascending) ▷ Sort in order of the likelihood
  $X[i] \leftarrow t_i X_{i-1}$ ▷ Shrink the prior volume
  ▷ t_i is a random variable with distribution $P(t) = Nt^{N-1}$
  $\mathcal{Z} \leftarrow \mathcal{Z} + \mathcal{L}(\text{livepoints}[0])w_i$
  do
    new_point ← draw_sample($\pi(\Theta)$, 1)
    if $\mathcal{L}$(livepoints[0]) < $\mathcal{L}$(new_point) **then**
      livepoints[0] ←new_point ▷ Discard the old point with the lowest likelihood
      exit
    end if
  end do
end while
$\mathcal{Z} \leftarrow \mathcal{Z} + \frac{X_i}{N} \sum_{j=1}^{N} (\mathcal{L}(\text{livepoints}[j]))$
▷ $P(t)$ is the probability distribution for the largest of N samples drawn uniformly. After i iterations the prior volume will be narrowed such that $\ln X_i \approx -(i \pm \sqrt{i})/N$. [8]

2.2.1 Nested Sampling

The relation between the likelihood and prior volume are exploited by nested sampling to transform the multidimensional evidence integral from Equation 2.2 into an one-dimensional integral. [12] The (differential) volume of the prior is defined by $dX = \pi(\Theta)d^D\Theta$, so that

$$X(\lambda) = \int_{\mathcal{L}(\Theta)>\lambda} \pi(\Theta)d^D\Theta, \tag{2.5}$$

with λ raised from ≈ 0 to 1 while navigating through the log-likelihood and where the integral extends over the regions of parameter space contained within the iso-likelihood contour $\mathcal{L}(\Theta) = \lambda$. Now we consider for a moment $\mathcal{L}(X)$ is a monotonically decreasing function, i.e. the inverse of Equation 2.5, then the evidence integral from Equation 2.2 is

$$\mathcal{Z} = \int_0^1 \mathcal{L}(X)dX. \tag{2.6}$$

With this equation one can evaluate the likelihoods $\mathcal{L}_i = \mathcal{L}(X_i)$, where X_i is a sequence of decreasing values,

$$0 < X_M < \cdots < X_2 < X_1 < X_0 = 1, \tag{2.7}$$

as illustrated in Figure 2.1. The evidence can be approximated numerically using standard quadratur methods as a weighted sum [12]

$$\mathcal{Z} = \sum_{i=1}^{M} \mathcal{L}_i w_i \tag{2.8}$$

By using the trapezium rule (see [27]) we get $w_i = \frac{1}{2}(X_{i-1} - X_{i+1})$.

As we stated before MultiNest is able to evaluate the evidence. For Equation 2.8 nested sampling performs the algorithm shown in Algorithm 1. Algorithm 1 does not explicitly state when to stop. Skilling provides in [30] a robust stopping criterion by estimating the possible gain from further iterations. One takes the largest likelihood $\mathcal{L}_{\max}$ of the current active set of points and calculates $\Delta\mathcal{Z}_i = \mathcal{L}_{\max} \cdot X_i$ which shall be greater than some user-defined evidence tolerance factor. In log-evidence 0.1 is often used but we are going to give different values in Chapter 4.1 for each toy model.

Given the evidence $\mathcal{Z}$, one can calculate the posterior inferences by using the discarded points, i.e. the point with the lowest likelihood at each iteration. These points are assigned the weights

$$p_i = \frac{\mathcal{L}_i w_i}{\mathcal{Z}}$$

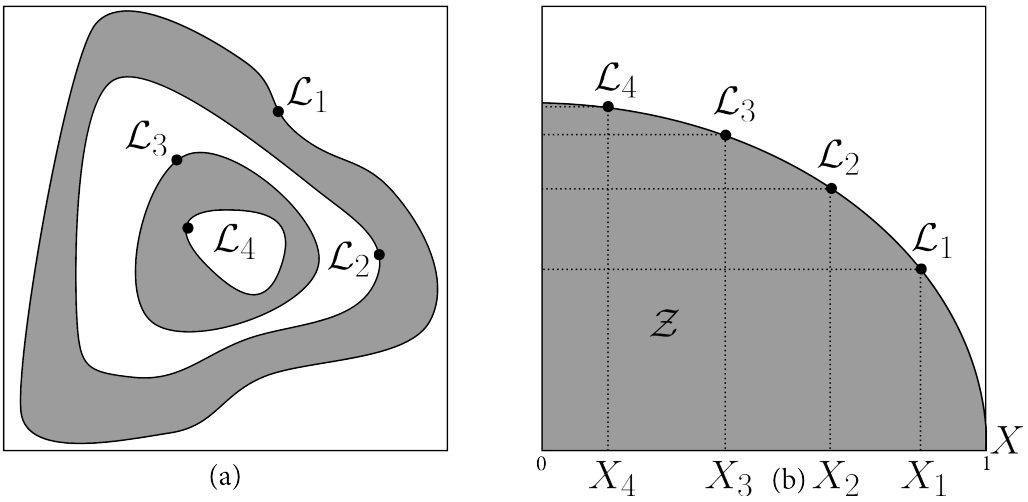


Fig. 2.1: (a) is an example for the posterior of a two dimensional problem. (b) is the transformed $L(X)$ function where the prior volumes X_i are associated with each likelihood L_i.

which can now be used to calculate inferences of posterior parameters such as means, standard deviations, covariances etc. (see [18] for more information about such calculations).

We present the evidence error estimation shortly. A full explanation can be found in [8]. The negative relative entropy H is given by

$$H = \int_0^1 \ln\left(\frac{\mathrm{d}P}{\mathrm{d}X}\right)\mathrm{d}X \approx \sum_{i=1}^{M} \frac{\mathcal{L}_i w_i}{Z} \ln\left(\frac{\mathcal{L}_i}{\mathcal{Z}}\right), \tag{2.9}$$

where P is the posterior. The evidence is given by

$$\ln \mathcal{Z} = \ln\left(\sum_{i=1}^{M} \mathcal{L}_i w_i\right) \pm \sqrt{\frac{H}{N}}. \tag{2.10}$$

Putting together Equation 2.9, 2.10 and the error of the discretisation in Equation 2.8, the error will be $\mathcal{O}(1/M^2)$ (see [30] at page 836-837).

2.2.2 Ellipsoidal Nested Sampling

The algorithm above shall now be improved by using ellipsoids as approximation for the iso-likelihood contour. Instead of drawing blindly from the prior within the hard constraint $\mathcal{L} > \mathcal{L}_i$ one builds the covariance matrix of the current set of live (also called active) points. [26] The iso-likelihood contour may not be exactly ellipsoidal and therefore one enlarges the ellipsoid by a factor f. A new point is then sampled within this ellipsoid. The acceptance rate tends to unity if the ellipsoid matches the true iso-likelihood-contour. For drawing uniform samples from a D-dimensional ellipsoid you may look at [29]. This method can easily be extended to non-uniform priors. In multimodal regions one ellipsoid cannot match the true iso-likelihood contour because in between two distinct modes one ellipsoid would enclose low likelihood regions, too.

2.2.3 Ellipsoidal Nested Sampling with Multiple Ellipsoids

Ellipsoidal nested sampling can be improved by using more ellipsoids. For that purpose one identifies distinct clusters of live points and constructs an ellipsoid for each cluster. Since the evidence is linear each cluster can be processed independently and the contributions can be summed up. One can assume that the prior volume and the number of points within each clustered region are proportional since the N active points are uniformly distributed.
In order to identify all the clusters and partition them correctly a special clustering algorithm needs to be used, which is called 'Dinosaur clustering' in the original code from [14]. We explain this part just as in [12]. First one uses an expectation maximization approach to find the optimal ellipsoidal decomposition of N active points. The N active points are denoted in the unit hypercube by $S = \{u_1, u_2, \ldots, u_N\}$. The partitioning of S into K clusters is labeled as $\{S_k\}_{k=1}^{K}$ with $K \geq 1$ and $\cup_{k=1}^{K} S_k = S$. A reasonably accurate and computationally simple way to calculate the minimum volume bounding ellipsoid for a cluster S_k is

$$E_k = \{u \in \mathbb{R}^D | u^T \cdot (f_k \cdot C_k)^{-1} \cdot u \leq 1\}, \tag{2.11}$$

where

$$C_k = \frac{1}{n_k} \sum_{j=1}^{n_k} (u_j - \mu_k) \cdot (u_j - \mu_k)^T. \tag{2.12}$$

C_k is an empirical covariance matrix of S_k, $\mu_k = \sum_{j=1}^{n_k} u_j$ is the center of the mass of the subset and n_k is the number of points in the subset S_k at the end of the i^{th}

iteration. The volume $V(E_k)$ of the ellipsoid is then proportional to $\sqrt{\det(f_k C_k)}$. If one minimizes the volume of all ellipsoids divided by the volume of the whole set

$$F(S) \equiv \frac{1}{V(S)} \sum_{k=1}^{K} V(E_k) \tag{2.13}$$

subject to $F(S) \geq 1$ and with respect to K-partitionings $\{S_k\}_{k=1}^{K}$ one obtains the optimal solution for the number of cluster K. Since $V(S)$ is not known, the minimization is done by using an 'expectation-minimization' scheme. This scheme uses the result of [24] which states that for uniformly distributed points, the variation in $F(S)$ from reassigning a point with position $u \in S_k$ to $S_{k'}$ is

$$\Delta F(S)_{k,k'} \approx \gamma \left(\frac{V(E_{k'}) \cdot d(u, S_{k'})}{V(S_{k'})} - \frac{V(E_k) \cdot d(u, S_k)}{V(S_k)} \right) \tag{2.14}$$

with γ a constant and the Mahalanobis distance from u to the center μ_k of E_k from Equation 2.11

$$d(u, S_k) = (u - \mu_k)^T \cdot (f_k \cdot C_k)^{-1} \cdot (u - \mu_k). \tag{2.15}$$

The true volume from which S_k has been drawn is

$$V(S_k) = \frac{n_k \cdot V(S)}{N}. \tag{2.16}$$

Equation 2.16 is used to force the constraint $V(E_k) \geq V(S_k)$ by enlarging E_k by f_k such that $V(E_k) = \max[V(E_k), V(S_k)]$ before calculating Equation 2.13 and 2.14. In this case $V(S)$ corresponds to the true remaining prior volume X_i which is unknown but the expectation value of this random variable is known. In order to define $V(S_k)$ as per Equation 2.16 one takes $V(S) = \exp(-i/N)$. Equation 2.13 can be minimized using Equation 2.14 with

$$h_k(u) = \frac{V(E_k) \cdot d(u, S_k)}{V(S_k)}, \tag{2.17}$$

where $u \in S$ is assigned to $S_{k'}$ if and only if $h_k(u) < h_{k'}(u), \forall k \neq k'$.
Now all the tools to determine the optimal number of ellipsoids K are created and can be used recursively as in Algorithm 2 which starts with $K = 2$, optimizes this 2-partition with Equation 2.17 and recursively partitions the resulting ellipsoids.
As stated by [24], the minimizer can be over-conservative and the partition should still go on if the ellipsoidal volume is greater than the true volume by some factor (in Algorithm 2 in line 16 we used 2). The computation of eigenvalues and eigenvectors for ellipsoids is expensive with O(n^ω) where $\omega < 2.376$ [5] but MulitNest can avoid part of the partitioning algorithm during the nested sampling. With the results from Algorithm 2 MultiNest scales the ellipsoids at subsequent iteration $k + 1$ such that $V(E_{k+1}) = \max[V(E_k), X_{i+1} \cdot (n_k/N)]$ with X_{i+1} the remaining prior volume in the next nested sampling iteration. In later iterations of nested sampling, this scaling

becomes less effective, therefore Algorithm 2 is used if $F(S) \geq h$ with typically $h = 1.1$.
With this nested sampling application the ellipsoids might not enclose the entire iso-likelihood contour because the ellipsoidal approximation to a region in the prior space might not match exactly. Thus sampling can be done from a region which is greater than the prior volume by using X/e as minimum volume in Algorithm 2 with e the desired sampling efficiency. For undersampling one can set e to be greater than unity. Regardless of e, the ellipsoids E_k always enclose the subsets S_k.
After that one has K ellipsoids at iteration i and MultiNest chooses one ellipsoid to sample from determined by the volume fraction with probability

$$p_k = \frac{V(E_k)}{V_{\text{tot}}}, \tag{2.18}$$

where $V_{\text{tot}} = \sum_{k=1}^{K} V(E_k)$. Many of these ellipsoids are overlapping and a sampled point within n_e ellipsoids must choose an ellipsoid with probability n_e^{-1}.
In highly multimodal regions MultiNest needs a large number N of active points (compare Chapter 4.1.3) in order to detect all modes. But the more active points there are the more decreases the convergence rate. To compensate this MultiNest decreases the number of active points with further iterations such that the number of active points N_i at the i^{th} is

$$N_i = N_{i-1} - N_{\min} \frac{\Delta\mathcal{Z}_{i-1} - \Delta\mathcal{Z}_i}{\Delta\mathcal{Z}_i - \text{tol}}, \tag{2.19}$$

where $\Delta\mathcal{Z}_i = \mathcal{L}_{\max} X_i$ is the largest evidence contribution, tol is the tolerance on the final evidence used in the stopping criterion and $N_{\min} \leq N_i \leq N_{i-1}$ is the minimum number of active points allowed.
At last we want to note that the implementation of MultiNest distinguishes between isolated cluster (the initial set S which starts Algorithm 2) and subcluster (the resulting S_k from Algorithm 2). Each isolated cluster contains at least one subcluster with its covariance matrix saved. This differentiation will be used in Chapter 3.3.

Algorithm 2 calculate_ellipsoid(S)

$E_0 \leftarrow$ calculate_ellipsoid(S)
$V_0 \leftarrow$ volume_of(E_0)
$V \leftarrow$ volume_of(S)
$S_1, S_2 \leftarrow$ k-means(S) ▷ Partition the set in $K = 2$ sets
do
  for i=1,2 **do**
    $E_i \leftarrow$ calculate_ellipsoid(S_i)
    $V_i \leftarrow$ volume_of(E_i)
    $E_i \leftarrow$ enlarge(E_i) ▷ Enlarge such that $V(E_i) \leftarrow \max[V(E_i), V(S_i)]$
    $V_i \leftarrow$ enlarge(V_i)
  end for
  for all $u \in S$ **do**
    assign(u, S_1, S_2) ▷ Assign to S_k such that $h_k(u) \leftarrow \min[h_1(x), h_2(x)]$
  end for
while at least one point has been reassigned
if $V_1 + V_2 < V_0$ **or** $V_0 > 2V$ **then**
  $E \leftarrow$ calculate_ellipsoid(S_1) ▷ E is here a list of all ellipsoids
  $E \leftarrow$ concatenate(E, calculate_ellipsoid(S_2))
  return E ▷ E is a list of optimal ellipsoids for the sets S_1 and S_2
else
  return E_0 ▷ E_0 is the optimal ellipsoid for the set S
end if

2.2.4 A short pseudocode for MultiNest

Algorithm 3 MultiNest(N)

$Z \leftarrow 0$ ▷ Initialize the evidence
livepoints←draw_samples($\pi(\Theta)$, N) ▷ Draw N samples from full prior
ic← createCluster(livepoints) ▷ Create one isolated cluster with all points
use_subclustering = false
for $i = 1$,max_iterations **do**
  if necessary **then**
    detect_modes()
  end if
  if use_subclustering **then**
    for each ic **do**
      calculate_ellipsoid(livepoints_in_ic) ▷ See Algorithm 2
    end for
  end if
  for $j = 1$, number of isolated cluster **do**
    if ic_j is not finished **then**
      adjust_prior_volumes(ic_j)
      lowest_point←lowest_likelihood_point(ic_j)
      if there is only one subcluster in total **then**
        do
          new_point←sample(ic_j)
          if $\mathcal{L}$(new_point) $>$ $\mathcal{L}$(lowest_point) **then**
            exit
          end if
        end do
      else
        do
          sc_k ←select_subcluster(ic_j) ▷ See Equation 2.18
          new_point←sample($\mathrm{subcluster}_k$)
          if $\mathcal{L}$(point) $>$ $\mathcal{L}$(lowest_point) **then**
            exit
          end if
        end do
        evolve_ellipsoids() ▷ Removes the old point and adds the new found point
      end if
      update_evidence()
    end if
  end for
  if acceptance_rate $<$ efficiency **then** ▷ The acceptance rate is the amount of
    ▷ needed samples until the hard constraint is satisfied
    use_subclustering←true
  end if
end for

3 Parallelization

> *There are 3 rules to follow when parallelizing large codes.*
> *Unfortunately, no one knows what these rules are.*
>
> **— W. Somerset Maugham, Gary Montry**

As discussed in Chapter 2.2 MultiNest is used to sample more efficiently and to avoid likelihood evaluations which do not help marching through the log-likelihood. Given Amdahl's law [19] for speedup

$$\Psi \leq \frac{1}{f + \frac{(1-f)}{p}}, \tag{3.1}$$

where $f \in [0, 1] \subset \mathbb{R}$ is the fraction of the algorithm, which is done sequentially and $(1 - f)$ is the fraction which can potentially be parallelized with p processors or threads, we have an upper bound for a possible speedup. Calculating the likelihood in each iteration is by far the biggest fraction (e.g. calculating the energy loss of muons in IceCube as we mentioned in Chapter 1) and therefore we focus on parallelizing this part.

In this section we present three different parallelization schemes and discuss the possible speedup and problems with each of them. The algorithms 4, 5, and 6 follow the original implementation from MultiNest v3.9 [14]. An isolated cluster is finished if the difference between the lowest and the highest likelihood is lower than or equal to 0.0001 or the possible contribution (as explained in Chapter 2.2.3) is smaller than a specified tolerance. The variable 'max_iterations' is userdefined and in this thesis just the largest possible 32 bit integer (2147483647).

We present different approaches to point out the difficulty of parallelizing MultiNest compared with other Monte Carlo methods (compare [22] and [28] for easy Monte Carlo cases) which do not suffer from degrading sampling efficiency or which do not explore concave or not differentiable regions.

3.1 1st Approach

Our first approach tries to reach the highest likelihoods faster by getting the best possible new point in each iteration. Every thread draws a new point for the current point with the lowest likelihood and the new point with the highest likelihood is used for further iterations while all the other points are discarded. A short pseudocode for this approach is given in Algorithm 4. The sampling efficiency is calculated by taking the number of needed calls to evaluate the likelihood for a sampled point divided by the number of used threads.

Algorithm 4 1st approach (an excerpt of Algorithm 3 beginning at line 17)

```
1:  if there is only one subcluster in total then
2:    do
3:      new_points[id]←sample(ic_j) in parallel        ▷ Each thread has its own id
4:      best_point←max(L(new_points))
5:      if L(best_point > L(lowest_point) then
6:        exit
7:      end if
8:    end do
9:  else
10:   do in parallel
11:     sc_k ←select_subcluster(ic_j)                  ▷ See Equation 2.18
12:     new_points[id]←sample(subcluster_k)            ▷ Each thread has its own id
13:     if id== 1 then accepted_point←lowest_point
14:       for point ∈ new_points do
15:         if L(point) > L(lowest_point) and L(point) > L(accepted_point) then
16:           Accept with probability 1/n_e            ▷ See Chapter 2.2.3
17:           if Accepted then
18:             accepted_point←point
19:           end if
20:         end if
21:       end for
22:     end if
23:     if Any point has been accepted then
24:       exit
25:     end if
26:   end parallel do
27:   evolve_ellipsoids()        ▷ Removes the old point and adds the new found point
28: end if
```

3.1.1 Discussion

On the one hand this approach can easily be implemented in the original code and sampling a new point is done more efficiently especially for hard likelihood contours (compare with Gaussian shell with 10 dimensions in Chapter 4.1.2) and therefore the convergence rate is higher. On the other hand a linear speedup cannot be expected. The new drawn point might be better compared to a new point if it is only drawn with one thread but new points are drawn uniformly from the (current) prior. The new point is not always good enough to spare another iteration where the prior is shrinked down making it more likely to sample from high likelihood regions.
In this approach we also calculate the likelihood for some points which are discarded without using the information. A similar approach is already implemented in the original code [14] with MPI where the discarded points' information are used via importance nested sampling which assigns each point a weight for calculating the posterior more accurately. The original implementation also does not just take the best point, it rather takes the first point which satisfies the hard constraint and uses the points which have been discarded until now for importance nested sampling. The remaining sampled points are used in the next iteration until no point remains before every thread samples a new point. [11] This (pseudo-)importance nested sampling is just an alternative summation of drawn points which leads to a better accuracy for the Bayesian evidence without changing the way MultiNest explores the parameters space. Our approach on the other side uses the original exploration method but parallelized with fewer OpenMP instructions than MPI instructions generating less overhead in this certain algorithm. In general MPI and OpenMP deliver a similar parallel performance. [2]

3.2 2nd Approach

The second approach shall extend the information gain by each new point by assigning each thread a different point with a low likelihood from one isolated cluster, where the first thread has to sample a new point with a higher likelihood than the the lowest likelihood, the second thread needs a higher likelihood than the second lowest likelihood and so on (see Algorithm 5). The sampling efficiency is determined by taking the maximum amount of needed calls to evaluate the likelihood for a sampled point for all threads which leads to the earliest possible usage of clustering the current livepoints and fitting them with ellipsoids with Algorithm 2 (e.g. earlier than taking the mean of trials to sample a new point within the hard constraint $\mathcal{L}_i > \mathcal{L}$).

Algorithm 5 2nd approach (a short version of Algorithm 3 beginning at line 17)

```
1:  if there is only one subcluster in total then
2:     do in parallel                                   ▷ All variables here are private
3:        lowest_point←get_lowest(ic_j, id)             ▷ Each thread gets the id^th lowest point
4:        new_point←sample(ic_j)
5:        if L(new_point > 𝓛(lowest_point) then
6:           exit
7:        end if
8:     end parallel do
9:  else
10:    do in parallel                                   ▷ All variables here are private as above
11:       lowest_point←get_lowest_point(ic_j, id)
12:       sc_k = select_subcluster(ic_j)                ▷ See Equation 2.18
13:       new_point←sample(subcluster_k)
14:       if L(new_point) > 𝓛(lowest_point) then
15:          Accept with probability 1/n_e              ▷ See Chapter 2.2.3
16:          if Accepted then
17:             accepted_points[id]←new_point
18:             exit
19:          end if
20:       end if
21:    end parallel do
22:    for point ∈ accepted_points do
23:       evolve_ellipsoids()                           ▷ Removes the old point and adds the new found point
24:    end for
25: end if
```

3.2.1 Discussion

Here we would expect a linear speedup for the parallel fraction if we could stop the decreasing acceptance rate for many threads with decreasing prior volume and increasing likelihood. This approach is nearly as bad as drawing blindly from the prior. All in all we discard a great amount of the benefits given by ellipsoidal sampling as described in [26] and in Chapter 2.2.3. With more threads it gets more difficult to sample a point inside the hard constraint for threads with high ids.

3.3 3rd Approach

To overcome the disadvantages of the second approach, we choose to assign each thread a different isolated cluster. Within these clusters, each thread evaluates its point with the lowest likelihood and exchanges it accordingly. In further iterations with many modes, the first few clusters might get very small and thus not contribute very much to the evidence evaluation or might even be inactive. Clusters are assigned to threads in an interleaved pattern in order to distribute the workload for every thread as evenly as possible.

3.3.1 Discussion

Now every thread should be able to sample a new point within the hard constraint without a greater decreasing efficiency as in the original algorithm since each thread has its own isolated cluster to sample from. This is the most promising approach but it still has some problems. In applications with only few modes and therefore only few isolated clusters there might be some threads without assigned work. Even with many modes there might be some threads which have less unfinished isolated clusters than others since the clusters are not dynamically assigned to threads. This could be avoided in future implementations by adding global booleans for each isolated cluster which is true if the isolated cluster is free. With this information each thread could handle the next free isolated cluster. At last there is a slightly greater overhead for calculating the right indices for the points and a lot of barriers compared with the other approaches (see Algorithm 6).

Algorithm 6 3rd approach (a short version of Algorithm 3 beginning at line 13)

```
1: do in parallel
2:    ic_j ←get_next_ic()                                  ▷ Interleaved distribution
3:    if ic_j is not finished then                         ▷ Isolated cluster = ic
4:       !$OMP CRITICAL
5:       adjust_prior_volumes(ic_j)
6:       lowest_point←lowest_likelihood_point(ic_j)
7:       !$OMP END CRITICAL
8:       if there is only one subcluster in total then
9:          do
10:            new_point←sample(ic_j)
11:            if L(new_point > L(lowest_point) then
12:               exit
13:            end if
14:         end do
15:      else
16:         do
17:            sc_k ←select_subcluster(ic_j)              ▷ See Equation 2.18
18:            new_point←sample(subcluster_k)
19:            if L(point) > L(lowest_point) then
20:               exit
21:            end if
22:         end do
23:         !$OMP CRITICAL
24:         evolve_ellipsoids()   ▷ Removes the old point and adds the new found point
25:         !$OMP END CRITICAL
26:      end if
27:   end if
28:   !$OMP BARRIER
29:   if ic_j is not finished then
30:      !$OMP CRITICAL
31:      update_evidence()
32:      !$OMP END CRITICAL
33:   end if
34:   !$OMP BARRIER
35:   if id == 1 then
36:      finished←check_for_exit()                         ▷ Check if all ic are done
37:   end if
38:   !$OMP BARRIER
39:   if finished then
40:      exit
41:   end if
42: end parallel do
```

Applications & Implementation

4

> *In the first place, the best way to convey to the experimenter what the data tell him about θ is to show him a picture of the posterior distribution.*
>
> **— G. E. P. Box & G. C. Tiao**
> Bayesian Inference in Statistical Analysis (1973)

4.1 Applications

There are many different applications (see [25]) which can be used as test functions for maximation expectation algorithms like MultiNest. In this chapter we present three different toy models with multimodal regions, which have been used like that or with small modifications in various papers ([11], [9], [12], [8], [15], [30]). These toy models can be evaluated analytically hence giving precise evidence which can be compared to the result of MultiNest. The likelihood for a point is calculated very fast and therefore we are going to compare the sampling efficiency and the number of needed iterations from MultiNest in addition to the conventional speedup in terms of time. In order to give an idea for the speedup in real world problems (see Chapter 1) we added a call to `sleep()` for each likelihood evaluation such that the likelihood evaluation needs a bigger fraction of the overall runtime.

4.1.1 Himmelblau's Function

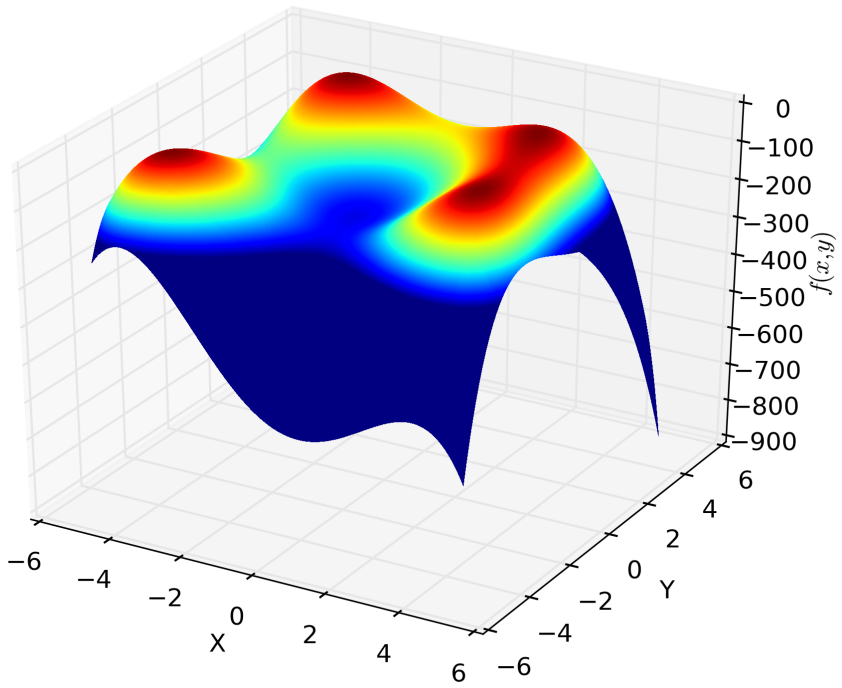


Fig. 4.1: This is a three-dimensional plot of Himmelblau's function with the likelihood on the z-axis. The red region indicates a higher value.

Himmelblau's function is defined as following:

$$f(x,y) = (x^2 + y - 11)^2 + (x + y^2 - 7)^2 \tag{4.1}$$

The local minima are at
$f(3,2) = 0$
$f(-2.805118, 3.131312) = 0$
$f(-3.779310, -3.283186) = 0$
$f(3.584428, -1.848126) = 0.$
We multiplied Himmelblau's function with -1 in order to get maxima instead of minima. This way the function looks like in Figure 4.1 and the contour of the likelihood looks like in Figure 4.2. Note that this is just a two dimensional function. Many real world problems have more parameters to evaluate and therefore they are harder to explore. This application has four modes with a steadily increasing likelihood once the four modes are detected which makes it a good test function for MCMC methods, too. The used variables for Himmelblau's function in MultiNest are listed in Table 4.1. The original MultiNest without a call to `sleep()` needs $t_{\text{total}} = 0.274415165$ s with $t_{\mathcal{L}} = 0.01421999$ s (the mean of 100 executions on the test system from Table 5.1). Therefore we can safely assume that $t = t_{\text{total}} - t_{\mathcal{L}} = 0.260195175$ s is the time

needed for MultiNest without the calculation of the likelihood.
With the longer likelihood evaluation MultiNest needs $t'_{\text{total}} = 16.8095589s$ (the mean of 100 executions on the test system from Table 5.1). The fraction for the likelihood evaluation with a call to `sleep()` is therefore $1 - t/t'_{\text{total}} \approx 98.4521\% = 1 - f$. With Amdahl's law from Equation 3.1 one can expect, at most, a speedup of $\Psi \leq 1/(f + (1-f)/p) = 1/(0.015479 + 0.984521/p) = 10.2541$ with $p = 12$ or $\Psi \leq 5.56899$ with $p = 6$.

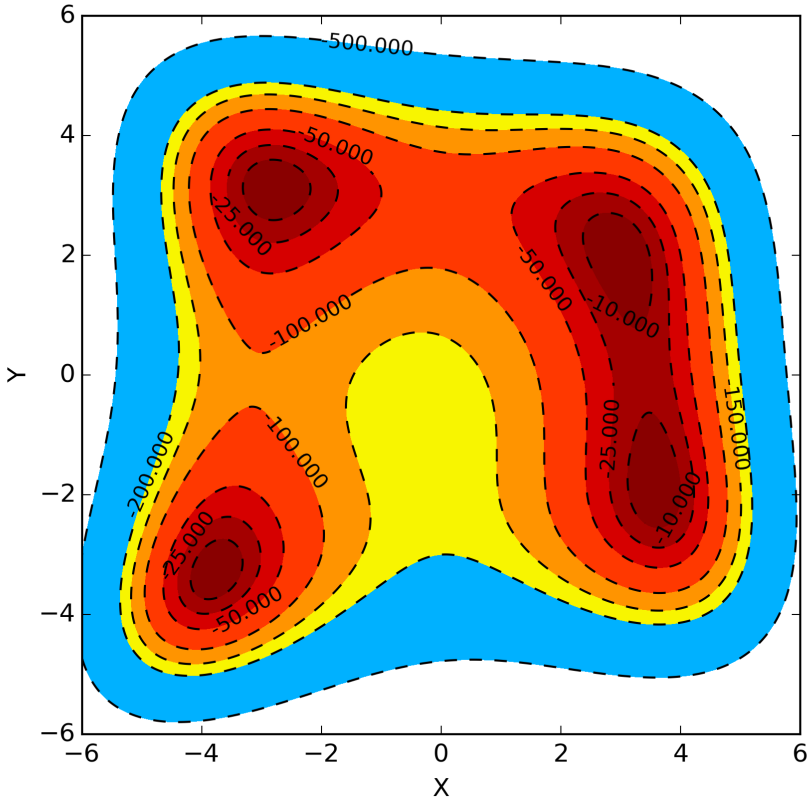


Fig. 4.2: This outlines the contour of Himmelblau's function. A similar presentation of the function will be used in Chapter 5.1 for comparison.

Tab. 4.1: Variables for Himmelblau's function

Sleep-timer	1000000 ns
Dimensions	2
Number of live points	500
Prior edges (all dimensions)	$(-6, 6)$
Evidence tolerance factor	0.001
enlargement factor f_k	0.5
Null evidence	$-1 \cdot 10^9$ (used for unknown null evidence)
Number of parameters to cluster	2

4.1.2 Gaussian Shell

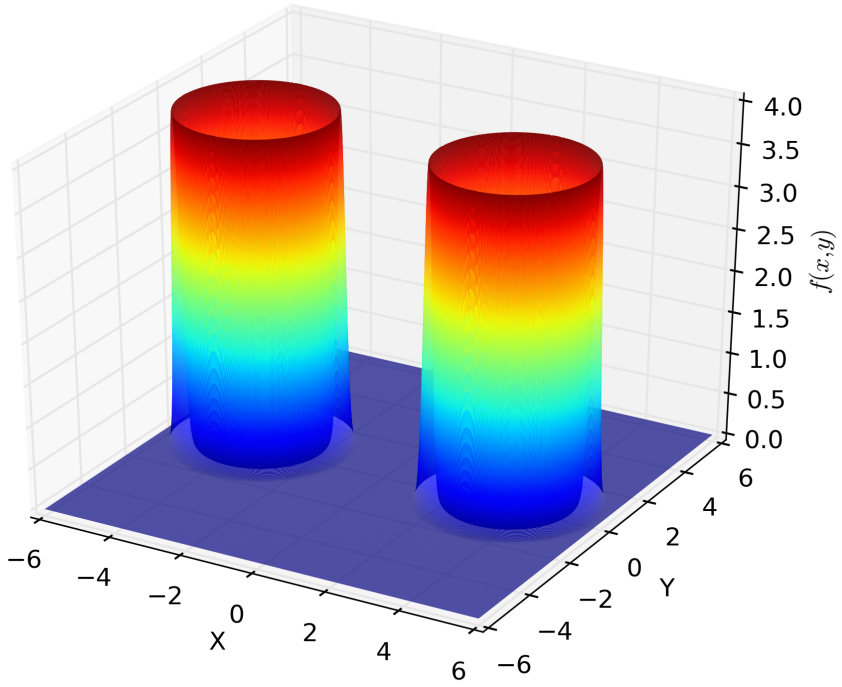


Fig. 4.3: This is a three-dimensional plot of the Gaussian shell function in two dimensions with the likelihood on the z-axis. The red region indicates a higher value.

The Gaussian shell as used in [12] is defined as follows:

$$\mathcal{L} = \mathrm{circ}(\theta; c_1, r_1, w_1) + \mathrm{circ}(\theta; c_2, r_2, w_2), \tag{4.2}$$

where

$$\mathrm{circ}(\theta; c, r, w) = \frac{1}{\sqrt{2\pi w^2}} \exp\left(-\frac{(|\theta - c| - r)^2}{2w^2}\right). \tag{4.3}$$

This results in two rings (see Figure 4.3), centred on c_1 and c_2 with radius r and width w. This function can be easily used in higher dimensions and it still provides different separated maxima. This function is picked because the distribution is representative which might occur in particle physics experiments in the context of 'beyond-the-Standard-Model paradigms' (see [12]) with small w. In Chapter 5.2 we are going to use a two- and ten-dimensional parameter space Θ with the values $w_1 = w_2 = 0.1$ and $r_1 = r_2 = 2$ as in [12] for comparison sake. The radii are set equally in order to have equal volumes for both rings or otherwise in higher dimensions it would be nearly impossible to detect the smaller ring. With $w_1 = w_2 = 0.1$ we also have a very thin ring which makes the evaluation more difficult. The used variables for Gaussian shell in MultiNest are listed in Table 4.2. As in Himmelblau's function we measured the time $t_{2D,\mathrm{total}} = 8.9089999 \cdot 10^{-2}$ s and $t_{2D,\mathcal{L}} = 3.7709999 \cdot 10^{-2}$ s. With the call

Tab. 4.2: Variables for Gaussian shell

Sleep-timer	100 ns
Dimensions	2 & 10
Number of live points	300
Prior edges (all dimensions)	$(-6, 6)$
Evidence tolerance factor	0.001
enlargement factor f_k	0.05
Null evidence	$-1 \cdot 10^9$ (used for unknown null evidence)
Number of paramters to cluster	2

to `sleep()` we measured $t'_{2D,\text{total}} = 2.6187$ s. The fraction for the likelihood fraction with a call to `sleep()` is therefore $1-(t_{2D,\text{total}}-t_{2D,\mathcal{L}})/t'_{2D,\text{total}} \approx 98.04\% = 1-f$. We apply Amdahl's law from Equation 3.1 and expect, at most, a speedup of $\Psi \leq 9.872$ with $p = 12$ or $\Psi \leq 5.464$ with $p = 6$.
Likewise in ten dimensions we measured $t_{10D,\text{total}} = 0.86019999$ s and $t_{10D,\mathcal{L}} = 0.52633$ s. With the call to `sleep()` we have $t'_{10D,\text{total}} = 14.62039999$ s and the fraction for the likelihood call is $1 - (t_{10D,\text{total}} - t_{10D,\mathcal{L}})/t'_{10D,\text{total}} \approx 97.72\% = 1 - f$. We use again Amdahl's law and expect a speedup of $\Psi \leq 9.594$ with $p = 12$ or $\Psi \leq 5.386$ with $p = 6$.

4.1.3 Eggbox Function

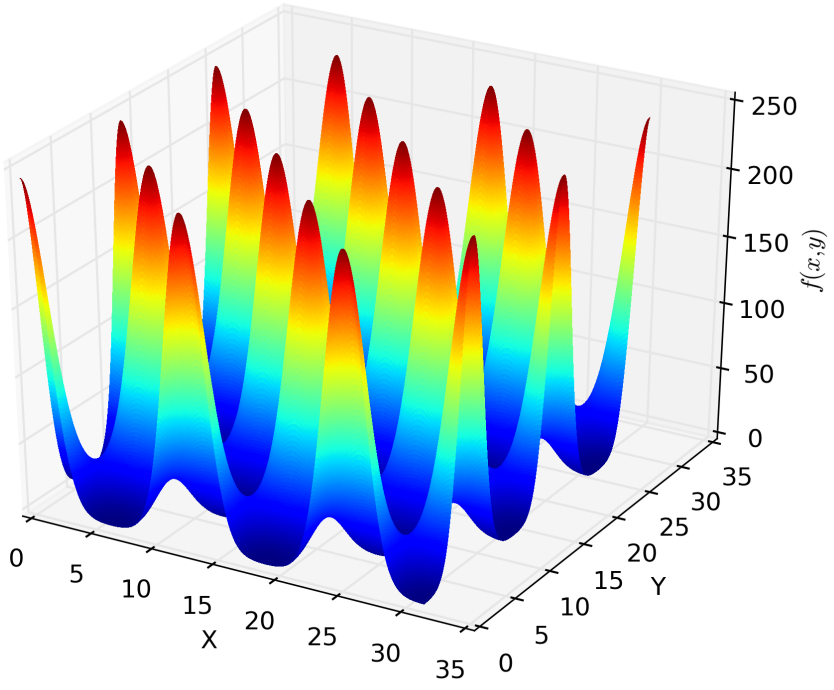


Fig. 4.4: This is a three-dimensional plot of the eggbox function with the likelihood on the z-axis. The red region indicates a higher value.

The eggbox function can be seen in Figure 4.4 and has been used as following:

$$f(x, y) = ((\cos(x/2) \cdot \cos(y/2)) + 2)^5 \tag{4.4}$$

This function is highly multimodal and has quite sharp edges. This distribution is similar to various astronomical object detection applications (see [12]) as mentioned in Chapter 1. The maxima are at

$$f(4\pi n_1, 4\pi n_2) = 243 \quad \text{with} \quad n_1, n_2 \in \mathbb{N}$$

and

$$f(4\pi n_1 + 2\pi, 4\pi n_2 + 2\pi) = 243 \quad \text{with} \quad n_1, n_2 \in \mathbb{N}.$$

We are going to use the function in the range of 0, 10π, which results in 18 maxima where 10 of them are exactly on the initial prior edge (compare Figure 4.5). These maxima are harder to enclose exactly by an ellipsoid. The used variables for the eggbox function in MultiNest are listed in Table 4.3. Like the other toy models before we measured $t_{total} = 0.897910178$ s, $t_{\mathcal{L}} = 3.101999 \cdot 10^{-2}$ s and $t' = 21.0310860$ s which gives the fraction of likelihood calls $1 - (t_{total} - t_{\mathcal{L}})/t' \approx 95.878\%$. By applying

Amdahl's law once again the best possible speedup is $\Psi \leq 8.2564$ with $p = 12$ and $\Psi \leq 4.9747$ with $p = 6$.

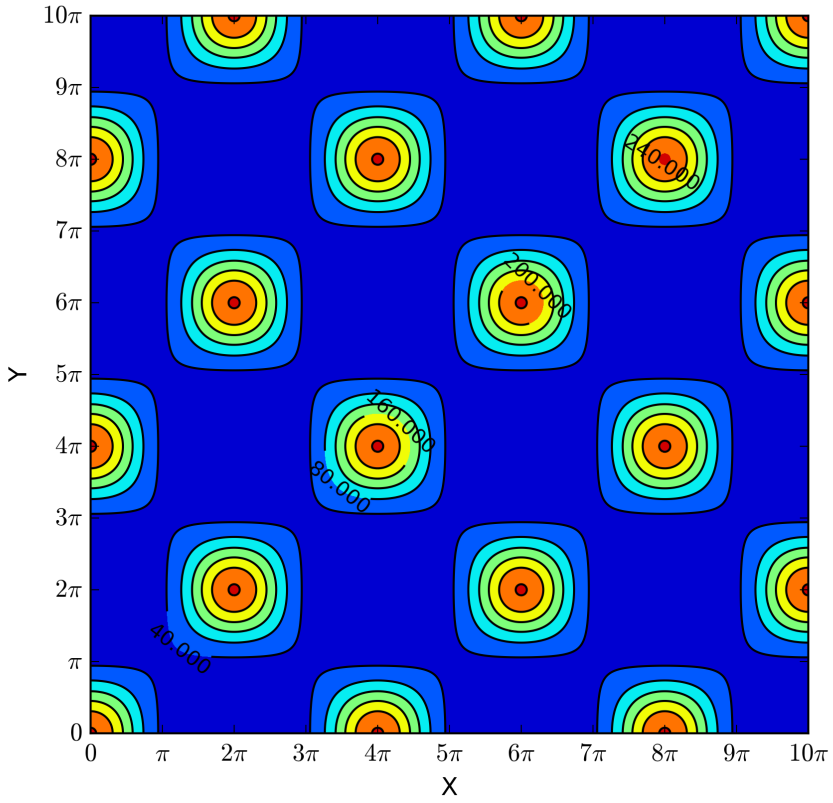


Fig. 4.5: This outlines the contour of the eggbox function. A similar presentation of the function will be used in Chapter 5 for comparison.

Tab. 4.3: Variables for eggbox function

Sleep-timer	500000 ns
Dimensions	2
Number of live points	2000
Prior edges (all dimensions)	$(0, 10 \cdot \pi)$
Evidence tolerance factor	0.5
enlargement factor f_k	0.5
Null evidence	$-1 \cdot 10^9$ (used for unknown null evidence)
Number of paramters to cluster	2

5 Performance & Scalability

> *Benchmarking—by which I mean any computer system that is driven by a controlled workload—is the ultimate in performance simulation. Aside from being a form of institutionalized cheating, it also offers countless opportunities for systematic mistakes in the way the workloads are applied and the resulting measurements interpreted.*
>
> **— Neil J. Gunther**
> Benchmarking Blunders and Things That Go Bump in the Night (2004)

In this chapter we are going to compare the runtime for each toy model with each approach. As mentioned in Chapter 4.1 we added a call to `sleep()` in each likelihood call in order to give a better idea of the possible speedup in real world problems. The implementation is based on MultiNest v3.9 which can be downloaded at [14]. The code is written in Fortran 95, therefore one has to be careful with indices of arrays since in Fortran arrays are stored in column-major order instead of row-major order like in other commonly used languages.
Throughout the performance evaluation we are going to use Pearson's correlation coefficient r since we assume the speedup and efficiency to be linear with the number of threads and the benchmarks to deliver normally distributed data.

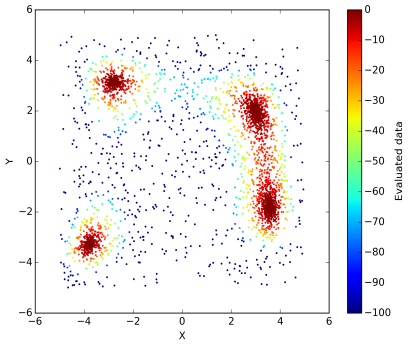
Processor	Intel® Core™ i7-3970X
Clock (GHz)	4
Cores	6 physical + 6 Hyperthreading
Cache (KB)	15360
main memory (GB)	32
main memory (type)	DDR3
Compiler	gfortran 4.8.4

Tab. 5.1: Experiment Setup

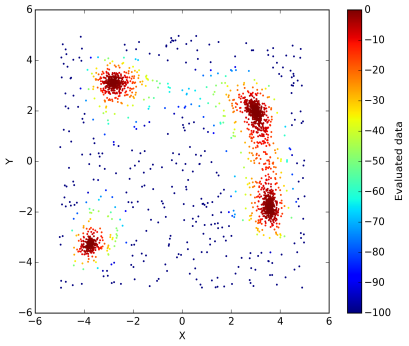
5.1 Himmelblau's Function

As you can see in Figure 5.2 the first approach gets faster up to the total number of physical cores+1 but throughout all amounts of threads the total speed is about the same as the sequential version. In Chapter 3.1.1 we explained that the first approach might converge faster to the highest likelihood which is not the case here. In Table 7.1 at Appendix A we see the reason for the lack of improvements. The sampling efficiency does not hold up with the number of threads ($r(\text{Samplingefficiency}, \text{Threads}) = 0.431$) and decreases compared to the sequential version. Even with less iterations used in MultiNest until the stopping condition is reached ($r(\text{Iterations}, \text{Threads}) = -0.9599$) the worse sampling efficiency annihilates the benefits of the first approach. The first approach also leads to a decreased accuracy for evaluating the evidence as you can see in Table 7.2 at Appendix B. The second approach is not listed anywhere because the sampling efficiency degrades so poorly that the program did not terminate after 10 minutes even without a call to `sleep()` in the likelihood function. This is even worse than expected and shows how important a good estimation of the prior with ellipsoids is in MultiNest.

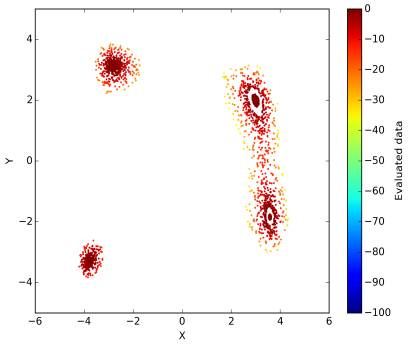
The third approach is clearly the best function and scales up to 6 threads which is the amount of physical cores in the testsystem as you can see in Figure 5.2. The speedup itself is not linear



(a) Sequential approach which explored the whole iso-likelihood contour with many points in uninteresting (blue) regions.



(b) 1st parallelization approach with 12 threads. The iso-likelihood contour is fully explored.



(c) 3rd parallelization approach with 12 threads. The regions with high likelihood are found very fast and thus nearly no point from the low likelihood region is used for the calculation of the evidence.

Fig. 5.1: The sampled data points for Himmelblau's function used for evaluating the Bayesian evidence after every 1000th iteration. You may compare this to Figure 4.2

since the best speedup is at 1.8 with 6 threads. In Chapter 3.3.1 we expected the speedup to correlate with the number of modes (here 4) but the test results state that the speedup grows with the amount of physical cores which is good for Himmelblau's function with such few total modes.
The efficiency for sampling and the total number of likelihood calls and iterations are listed in Table 7.1 at Appendix A. The sampling efficiency is about the same as in the sequential version as well as the number of iterations needed which is natural for this approach since the exploration of the iso-likelihood is not changed in the third approach in contrast to the first approach. The sampling efficiency also does not correlate very highly with the amount of threads ($r(\text{Samplingefficiency}, \text{Threads}) = -0.179$) which is also true for the number of iterations ($r(\text{Iterations}, \text{Threads}) = 0.081$).
Even with less points used from low likelihood regions (see Figure 5.1) the calculated evidence is as accurate as the sequential version (see Table 7.2 at Appendix B).

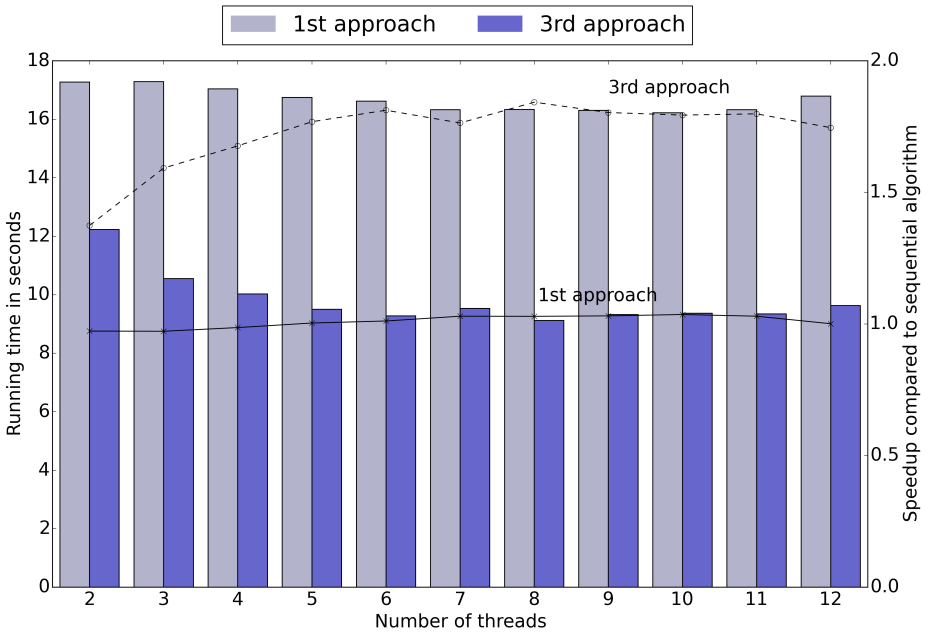
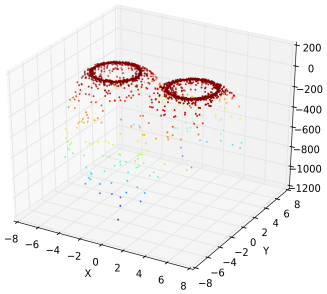


Fig. 5.2: Performance overview for Himmelblau's function. The line graphs, calibrated by the right y-axis, show the speed-ups obtained with each approach over the single threaded original version. A higher line indicates a greater speedup. The bar graphs belong to the left y-axis and show the running time. Both graphs are obtained by taking the mean after 100 executions.

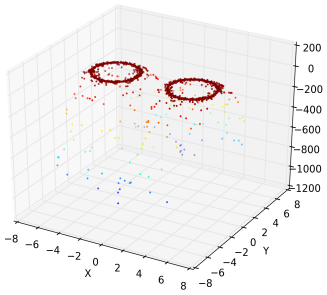
5.2 Gaussian Shell

In Figure 5.4 for the two dimensional Gaussian shell is the first approach always faster than the third approach with a speedup between 1.35 and 2.5 with 11 threads compared to a speedup from 0.95 to 1.34 with 8 threads. If you look at Figure 4.3 for a visualization of Gaussian shell you can see the distinct and narrow likelihood contour with a big flat region which does not contain any information for the high region. If a point is drawn in that flat region, then it does not help as much finding the mode as it does in Himmelblau's function or in the eggbox function. Due to that one has to increase the probability to sample from anywhere near a mode which is done by the first parallelization approach. The 3rd approach does not help very much since there are only two modes which results in few isolated clusters which is not helpful for the third approach (see Chapter 3.3.1). In fact one can even see a small 'speeddown' with 6 threads. This trend continues with higher dimensions where the first approach leads to a speedup of 1.51 and 3.06 with 10 threads compared to a speedup from 1.31 to 1.69 with 9 threads in the third approach. The speedup grows with more dimensions which is a result of the more difficult and narrow iso-likelihood contour in higher dimensions.

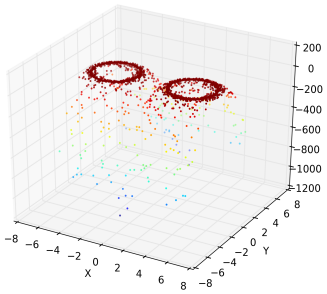
In Figure 5.3 we can see only small differences between all versions of MultiNest due to the narrow likelihood region so that the convergence rate of MultiNest is very high and the points cannot



(a) Sequential approach which explored a small part of the iso-likelihood contour.



(b) 1st parallelization approach with 12 threads.



(c) 3rd parallelization approach with 12 threads.

Fig. 5.3: The sampled data points for Gaussian shell in two dimensions used for evaluating the Bayesian evidence after every 100th iteration. You may compare this to Figure 4.3

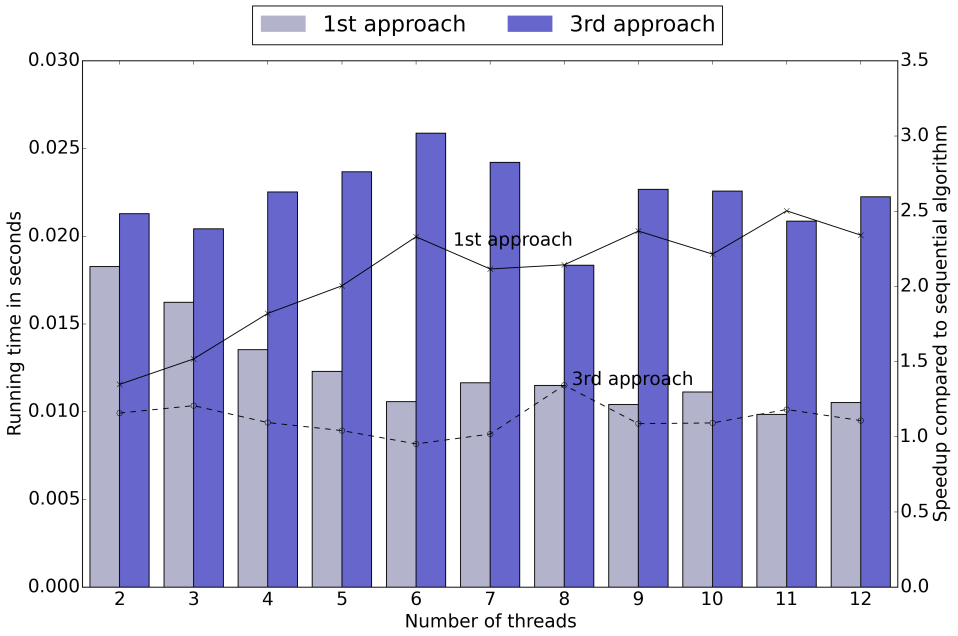


Fig. 5.4: Performance overview for Gaussian shell in two dimensions. The line graphs, calibrated by the right y-axis, show the speed-ups obtained with each approach over the single threaded original version. A higher line indicates a greater speedup. The bar graphs belong to the left y-axis and show the running time. Both graphs are obtained by taking the mean after 100 executions.

be distinguished in the upper rings anymore.

In Table 5.2 we can see almost the same results for the third approach as in Himmelblau's function. The sampling efficency is stable and as good as in the sequential version throughout all dimensions. The best values in the first approach are obtained by using as many threads as possible although 12 threads never delivered the best result. The evidence evaluation remains stable for two dimensions (Table 7.3 at Appendix C) and for ten dimensions (Table 7.4 at Appendix D). All in all the more dimensions are used for Gaussian shell the better the first approach scales.

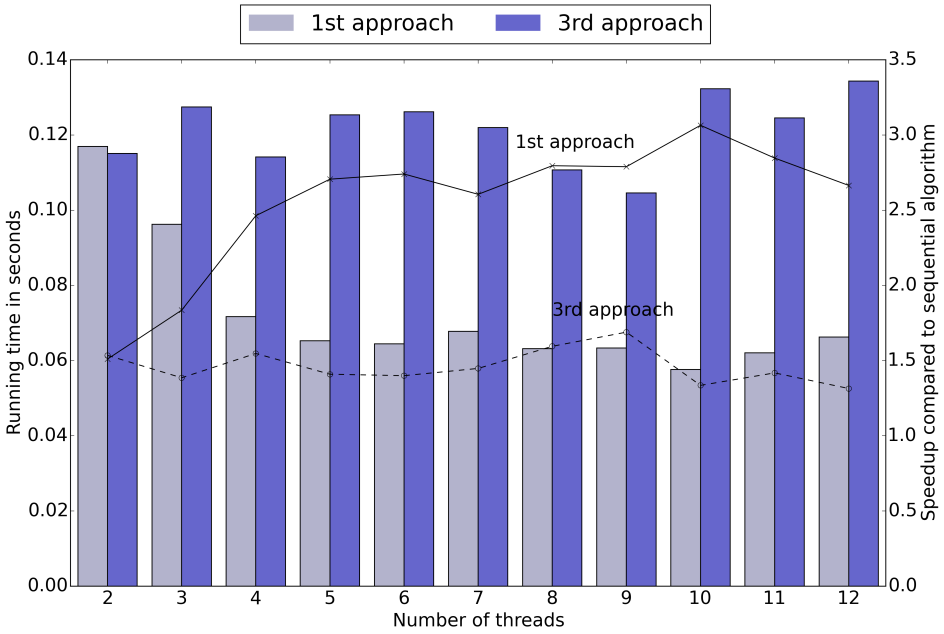


Fig. 5.5: Performance overview for Gaussian shell in ten dimensions. The line graphs, calibrated by the right y-axis, show the speed-ups obtained with each approach over the single threaded original version. A higher line indicates a greater speedup. The bar graphs belong to the left y-axis and show the running time. Both graphs are obtained by taking the mean after 100 executions.

Sampling efficiency			
D	sequential	1st approach	3rd approach
2	5.95%	17.31% (11)	7.40% (8)
5	2.95%	14.597% (11)	3.28% (3)
10	3.04%	11.15% (11)	3.31% (9)
20	3.58%	7.37% (11)	3.86% (5)
30	1.799%	4.29% (11)	2.00% (3)
Total iterations			
D	sequential	1st approach	3rd approach
2	1674.98	1454.80 (7)	1646.90 (5)
5	2426.55	1666.80 (11)	2371.80 (4)
10	4678.17	4002.199 (10)	4634.10 (6)
20	9645.95	8968.12 (11)	9552.5 (10)
30	14760.72	14181.299 (10)	14393.599 (2)

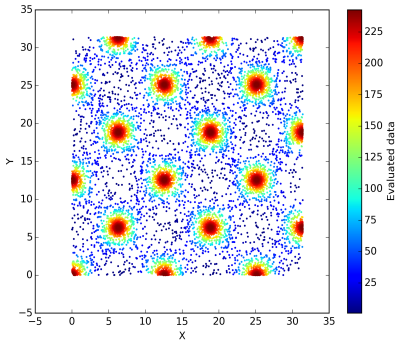
Tab. 5.2: The total number of iterations and the efficiency evaluated as explained in Chapter 3.1 and Chapter 3.3 for Gaussian shell subject to the dimensions D with 100 iterations per approach and dimension. The values are the best efficiency and the least amount of iterations where the amount of used threads are in brackets.

5.3 Eggbox Function

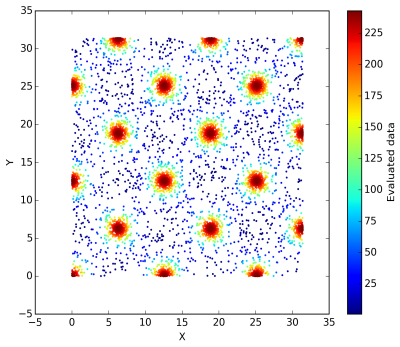
The first approach scales in the eggbox function even with the usage of hyper-threading saturating at a speedup between 1.41 and 1.44 with 9 to 12 threads as you can see in Figure 5.8. This is a consequence of the high peaks and the narrow likelihood contour where a luckily drawn point helps exploring the iso-likelihood contour faster as opposed to Himmelblau's function with its slowly increasing contour.

The second approach is not listed again due to the same reason why the first approach scales better than Himmelblau's function. The region is very thin and therefore a thread with an high id needs to find a point within the hard constraint $\mathcal{L} > \mathcal{L}_i$ but it samples within an ellipsoid E_k which is too big with many regions with $\mathcal{L} < \mathcal{L}_i$ for an efficiently sampling. That is why the second approach does not terminate.

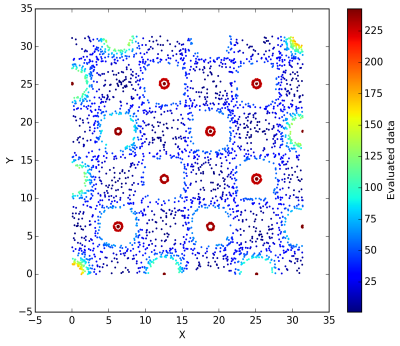
The third approach is the best version which scales up to 11 threads with a speedup of 2.15 just as expected in Chapter 3.3.1 although the total speedup is less than the estimated upper bound in Chapter 4.1.3. The many highly peaked modes result in many isolated cluster which are traversed in parallel resulting in a better scalability. In Chapter 5.1 Himmelblau's function scaled even with more threads than modes but such a statement cannot be made with the eggbox function because there are only 12 threads but 18 modes. The speedup of this function with more physical cores might be even higher.



(a) Sequential approach which explored the whole iso-likelihood contour.

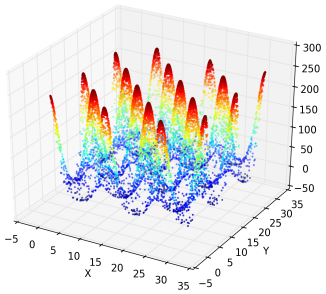


(b) 1st parallelization approach with 12 threads which explored the whole iso-likelihood contour with less points in low likelihood regions.

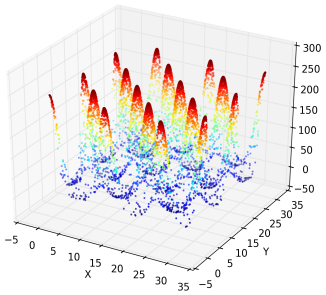


(c) 3rd parallelization approach with 12 threads. Some threads process their own cluster and add the contribution to the global evidence but the points are not added to the overall list of evaluated points, hence the empty spaces.

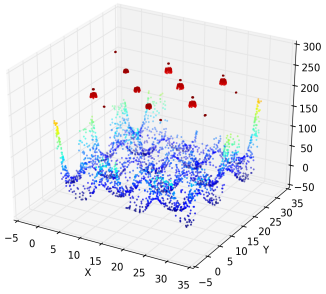
Fig. 5.6: The sampled data points for Gaussian shell in two dimensions used for evaluating the Bayesian evidence after every 1000th iteration. You may compare this to Figure 4.5



(a) Sequential approach which explored the whole iso-likelihood contour.



(b) 1st parallelization approach with 12 threads which explored the whole iso-likelihood contour with less points in low likelihood regions.



(c) 3rd parallelization approach with 12 threads. Some threads process their own cluster and add the contribution to the global evidence but the points are not added to the overall list of evaluated points, hence the empty spaces.

Fig. 5.7: The sampled data points for Gaussian shell in two dimensions used for evaluating the Bayesian evidence after every 1000th iteration. You may compare this to Figure 4.4

By looking at Table 7.5 at Appendix E we can see that the first approach needs less iterations with more threads ($r(\text{Iterations}, \text{Threads}) = -0.982$) and the sampling efficiency increases ($r(\text{Samplingefficiency}, \text{Threads}) = 0.973$). The third approach remains stable again with the number of iterations and the sampling efficiency ($r(\text{Iterations}, \text{Threads}) = 0.154$ and $r(\text{Samplingefficiency}, \text{Threads}) = 0.014$). If we compare the accuracy for local modes from Table 7.6 at Appendix F we see that the first approach has a slightly worse result than the sequential version while the third approach is slightly better than the sequential version. The third approach is here overall the best approach in terms of speedup, scalability and accuracy. In Figure 5.6 and Figure 5.7 we can see how the likelihood contour is explored although the third approach looks a bit off due to the distribution of isolated clusters to different threads. The gathered points for evidence estimation are overwritten by the last sampled point within one iteration by the other threads and thus not drawn. Despite that, the contribution of the overwritten point has been added to the global and local evidence.

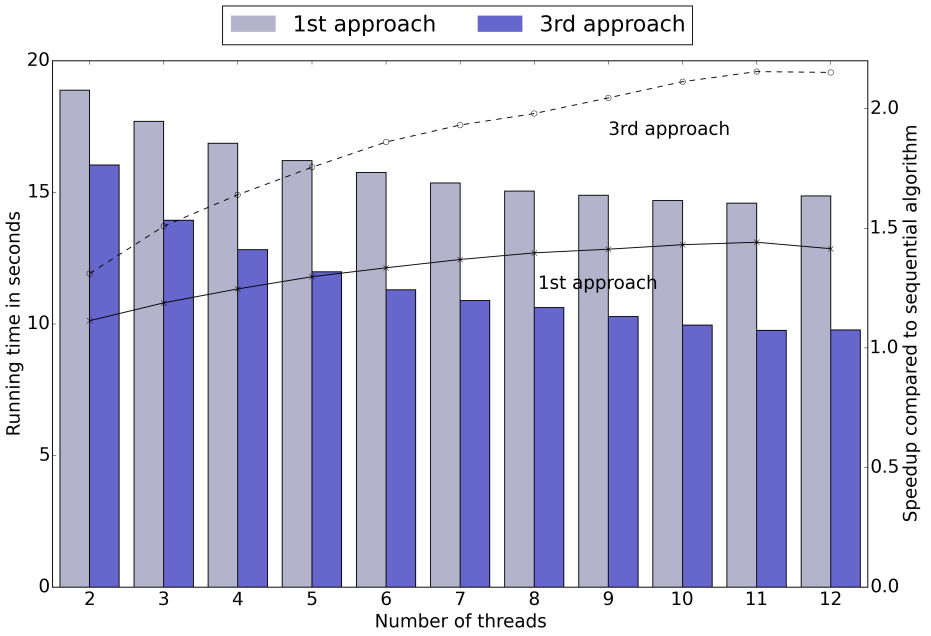


Fig. 5.8: Performance overview for the eggbox function. The line graphs, calibrated by the right y-axis, show the speed-ups obtained with each approach over the single threaded original version. A higher line indicates a greater speedup. The bar graphs belong to the left y-axis and show the running time. Both graphs are obtained by taking the mean after 100 executions.

6 Conclusion

> *Follow the evidence to where it leads, even if the conclusion is uncomfortable.*
>
> — **Steven James**
> The Knight

We presented three different parallelization approaches and three toy models with different results. As a conclusion of the second parallelization approach one has to use the benefits MultiNest delivers by shrinking the prior volume and sampling within ellipsoids or else the parallelization will not help or even worsen the algorithm. With respect to Gaussian shell and the first parallelization approach one can state that increasing the sampling efficiency further with the usage of many threads gives a big speedup for identifying high likelihood regions especially if these are very narrow and highly peaked, but on the other hand the eggbox function indicates a better speedup with the third parallelization approach if there are at least as many modes as threads available.
The best parallelization depends on the likelihood contour. In further work one could mix the first and the third parallelization approaches by assigning one isolated cluster more than one thread and then continue with the first approach. This could be used in a dynamic pattern where a decreasing sampling efficiency leads to the first approach with more threads assigned to one isolated cluster and the other way around.
In addition one could use GPUs which have thousands of cores (e.g. 3072 CUDA cores in GeForce GTX TITAN X). The downside of using a GPU is the bandwidth between the CPU and the GPU which is about 6 GB/s and the need to calculate the likelihood on the GPU which might be a challenge for itself due to the limited VRAM (e.g. 12 GB on the GeForce GTX TITAN X). I recommend to implement these approaches with importance nested sampling and different modes of MultiNest which usually have some small biased results and need more memory for calculating applications with 50 dimensions or more faster than the original sampling method. All in all one can use these parallelization approaches especially if the evidence calculation does not need to be as precise as possible and if the highest likelihood is more interesting. These approaches deliver a speedup between 1.5 and 2.8 with 6 physical cores on the CPU (and additional 6 via Hyperthreads) without the need to buy additional hardware like a GPU or more main memory.

7 Appendices

A Sampling efficiency of Himmelblau's function

Threads	approach	Sampling efficiency	Total iterations
1	Sequential	48.94%	2550.68
2	1st approach	44.23%	2413.11
	3rd approach	50.56%	2545.27
3	1st approach	43.15%	2313.26
	3rd approach	50.51%	2573.26
4	1st approach	42.81%	2276.55
	3rd approach	50.39%	2559.26
5	1st approach	42.91%	2250.56
	3rd approach	51.15%	2551.38
6	1st approach	42.77%	2186.189
	3rd approach	50.34%	2572.93
7	1st approach	42.77%	2183.419
	3rd approach	50.54%	2563.63
8	1st approach	43.00%	2140.87
	3rd approach	49.87%	2533.83
9	1st approach	43.68%	2108.32
	3rd approach	50.08%	2559.78
10	1st approach	43.799%	2120.96
	3rd approach	50.39%	2576.11
11	1st approach	44.27%	2093.53
	3rd approach	51.36%	2568.31
12	1st approach	44.17%	2071.28
	3rd approach	49.85%	2550.69

Tab. 7.1: The number of total likelihood calls and the efficiency evaluated as explained in Chapter 3.1 and Chapter 3.3 for Himmelblau's function as well as the total number of iterations. The values are the mean after 100 executions.

B Evidence evaluation of Himmelblau's function

Threads	approach	$\log(\mathcal{Z})$
1	Sequential	$-5.47084 \pm 9.44741 \cdot 10^{-2}$
2	1st approach	$-4.938798 \pm 8.95927 \cdot 10^{-2}$
	3rd approach	$-5.38862 \pm 9.36925 \cdot 10^{-2}$
3	1st approach	$-4.57062 \pm 8.57411 \cdot 10^{-2}$
	3rd approach	$-5.47488 \pm 9.44745 \cdot 10^{-2}$
4	1st approach	$-4.36782 \pm 8.34907 \cdot 10^{-2}$
	3rd approach	$-5.50862 \pm 9.47246 \cdot 10^{-2}$
5	1st approach	$-4.21406 \pm 8.17486 \cdot 10^{-2}$
	3rd approach	$-5.46364 \pm 9.43603 \cdot 10^{-2}$
6	1st approach	$-4.10858 \pm 8.06848 \cdot 10^{-2}$
	3rd approach	$-5.49175 \pm 9.47022 \cdot 10^{-2}$
7	1st approach	$-3.92253 \pm 7.83081 \cdot 10^{-2}$
	3rd approach	$-5.48247 \pm 9.454695 \cdot 10^{-2}$
8	1st approach	$-3.83407 \pm 7.73500 \cdot 10^{-2}$
	3rd approach	$-5.48361 \pm 9.46087 \cdot 10^{-2}$
9	1st approach	$-3.76233 \pm 7.63977 \cdot 10^{-2}$
	3rd approach	$-5.44269 \pm 9.41819 \cdot 10^{-2}$
10	1st approach	$-3.65764 \pm 7.52283 \cdot 10^{-2}$
	3rd approach	$-5.47534 \pm 9.45936 \cdot 10^{-2}$
11	1st approach	$-3.57519 \pm 7.43925 \cdot 10^{-2}$
	3rd approach	$-5.51192 \pm 9.48926 \cdot 10^{-2}$
12	1st approach	$-3.49576 \pm 7.34102 \cdot 10^{-2}$
	3rd approach	$-5.48235 \pm 9.46071 \cdot 10^{-2}$

Tab. 7.2: The log-evidence for Himmelblau's function evaluated using 100 iterations of MultiNest and taking the mean of all results.

C Evidence evaluation of Gaussian shell ($2D$)

Threads	approach	$\log(\mathcal{Z})$
-	Analytical	-1.75
1	Sequential	$-1.75016 \pm 9.37208 \cdot 10^{-2}$
2	1st approach	$-1.45965 \pm 8.86220 \cdot 10^{-2}$
	3rd approach	$-1.73102 \pm 9.34522 \cdot 10^{-2}$
3	1st approach	$-1.24772 \pm 8.48878 \cdot 10^{-2}$
	3rd approach	$-1.71162 \pm 9.30443 \cdot 10^{-2}$
4	1st approach	$-1.08034 \pm 8.17729 \cdot 10^{-2}$
	3rd approach	$-1.75033 \pm 9.36747 \cdot 10^{-2}$
5	1st approach	$-0.95036 \pm 7.92353 \cdot 10^{-2}$
	3rd approach	$-1.72636 \pm 9.333199 \cdot 10^{-2}$
6	1st approach	$-0.83598 \pm 7.70090 \cdot 10^{-2}$
	3rd approach	$-1.73459 \pm 9.33444 \cdot 10^{-2}$
7	1st approach	$-0.74153 \pm 7.52988 \cdot 10^{-2}$
	3rd approach	$-1.72609 \pm 9.31977 \cdot 10^{-2}$
8	1st approach	$-0.66427 \pm 7.37372 \cdot 10^{-2}$
	3rd approach	$-1.72022 \pm 9.33873 \cdot 10^{-2}$
9	1st approach	$-0.58796 \pm 7.22626 \cdot 10^{-2}$
	3rd approach	$-1.69277 \pm 9.27702 \cdot 10^{-2}$
10	1st approach	$-0.51829 \pm 7.081873 \cdot 10^{-2}$
	3rd approach	$-1.72329 \pm 9.335797 \cdot 10^{-2}$
11	1st approach	$-0.45476 \pm 6.96665 \cdot 10^{-2}$
	3rd approach	$-1.71818 \pm 9.31452 \cdot 10^{-2}$
12	1st approach	$-0.390199 \pm 6.83692 \cdot 10^{-2}$
	3rd approach	$-1.78189 \pm 9.41829 \cdot 10^{-2}$

Tab. 7.3: The analytical and estimated log-evidence for Gaussian shell in two dimensions. The analytical $\log(\mathcal{Z})$ is from [12]. The other values are results from different parallelization approaches and the sequential version of MultiNest.

D Evidence evaluation of Gaussian shell ($10D$)

Threads	approach	$\log(\mathcal{Z})$
-	Analytical	-14.59
1	Sequential	-14.54767 ± 0.22613
2	1st approach	-14.17111 ± 0.22324
	3rd approach	-14.63228 ± 0.22672
3	1st approach	-13.68549 ± 0.21986
	3rd approach	-14.45643 ± 0.22534
4	1st approach	-13.71174 ± 0.21998
	3rd approach	14.49202 ± 0.22574
5	1st approach	-13.51653 ± 0.21845
	3rd approach	-14.57195 ± 0.22635
6	1st approach	-13.28951 ± 0.21678
	3rd approach	-14.63232 ± 0.22685
7	1st approach	-13.05801 ± 0.21492
	3rd approach	-14.54202 ± 0.22609
8	1st approach	-13.11523 ± 0.21540
	3rd approach	-14.66471 ± 0.22694
9	1st approach	-12.867777 ± 0.21350
	3rd approach	-14.60901 ± 0.22664
10	1st approach	-12.82820 ± 0.21328
	3rd approach	-14.49539 ± 0.22579
11	1st approach	-12.5984 ± 0.21142
	3rd approach	-14.69453 ± 0.22717
12	1st approach	-12.63518 ± 0.21180
	3rd approach	-14.63058 ± 0.22673

Tab. 7.4: The analytical and estimated log-evidence Gaussian shell in ten dimensions. The analytical $\log(\mathcal{Z})$ is from [12]. The other values are results from different parallelization approaches and the sequential version of MultiNest.

E Sampling efficiency of the eggbox function

Threads	approach	Sampling efficiency	Total iterations
1	Sequential	51.35%	3345.37
2	1st approach	53.49%	3138.16
	3rd approach	53.36%	3323.199
3	1st approach	55.18%	3056.04
	3rd approach	54.38%	3361.699
4	1st approach	56.42%	3029.61
	3rd approach	53.66%	3346.50
5	1st approach	57.53%	2957.11
	3rd approach	53.81%	3340.80
6	1st approach	58.37%	2910.34
	3rd approach	53.19%	3346.699
7	1st approach	59.24%	2880.36
	3rd approach	54.01%	3342.50
8	1st approach	59.84%	2824.46
	3rd approach	53.82%	3304.30
9	1st approach	60.48%	2794.02
	3rd approach	53.71%	3300.60
10	1st approach	60.94%	2770.66
	3rd approach	53.12%	3353.199
11	1st approach	61.46%	2736.49
	3rd approach	54.26%	3390.199
12	1st approach	61.78%	2710.47
	3rd approach	53.74%	3346.82

Tab. 7.5: The number of total likelihood calls and the efficiency evaluated as explained in Chapter 3.1 and Chapter 3.3 for the eggbox function as well as the total number of iterations. The values are the mean after 100 executions.

F Local evidence evaluation of the eggbox function

Mode	true local $\log(\mathcal{Z})$	Sequential	1st approach	3rd approach
1	233.33	233.64 ± 0.23	234.86 ± 0.35	232.52 ± 0.24
2	233.33	227.53 ± 1.20	235.57 ± 0.20	232.29 ± 0.35
3	233.33	233.62 ± 0.23	235.54 ± 0.21	231.36 ± 0.41
4	233.33	232.38 ± 0.33	234.38 ± 0.54	232.39 ± 0.25
5	233.33	233.12 ± 0.24	234.73 ± 0.31	232.20 ± 0.26
6	233.33	233.06 ± 0.23	234.93 ± 0.29	232.13 ± 0.26
7	233.33	233.34 ± 0.23	235.62 ± 0.20	232.04 ± 0.26
8	233.33	233.51 ± 0.24	235.07 ± 0.28	232.59 ± 0.24
9	232.64	232.78 ± 0.33	235.48 ± 0.20	232.10 ± 0.35
10	232.64	231.38 ± 0.44	235.00 ± 0.29	232.19 ± 0.27
11	232.64	233.44 ± 0.22	235.72 ± 0.20	231.68 ± 0.38
12	232.64	233.45 ± 0.22	235.02 ± 0.32	231.69 ± 0.39
13	232.64	228.45 ± 0.55	235.58 ± 0.20	231.47 ± 0.42
14	232.64	232.74 ± 0.34	235.67 ± 0.20	231.46 ± 0.47
15	232.64	232.57 ± 0.37	235.19 ± 0.30	231.80 ± 0.37
16	232.64	232.53 ± 0.35	234.81 ± 0.29	231.21 ± 0.47
17	231.94	232.22 ± 0.39	235.59 ± 0.20	232.21 ± 0.26
18	231.94	233.10 ± 0.33	234.73 ± 0.44	231.64 ± 0.37

Tab. 7.6: The local log-evidence of each mode for the eggbox function. The true $\log(\mathcal{Z})$ is from [12] which has been calculated through numerical integration on a fine gride. The other values are results from different parallelization approaches of MultiNest with 12 threads.

Bibliography

[1]Kevin Abraham. *IceCube: neutrinos on graphics cards*. Technische Universität München, 2014 (cit. on p. 1).

[2]B. Armstrong, S.W. Kim, and R. Eigenmann. *Quantifying Differences between OpenMP and MPI Using a Large-Scale Application Suite*. http://link.springer.com/chapter/10.1007/3-540-39999-2_45. 2000 (cit. on p. 15).

[3]Döring Bortz. *Forschungsmethoden und Evaluation für Human- und Sozialwissenschaftler*. German. Vol. 4. Springer, 2006, p. 740 (cit. on p. 3).

[4]M. Bridges, F. Feroz and. M.P. Hobson, and A.N. Lasenby. *Bayesian optimal reconstruction of the primordial power spectrum*. http://mnras.oxfordjournals.org/content/400/2/1075.full.pdf+html. 2009 (cit. on p. 1).

[5]J. Demmel, I. Dumitriu, and O. Holtz. *Fast Linear Algebra is Stable*. http://arxiv.org/pdf/math/0612264v3.pdf. 2007 (cit. on p. 9).

[6]M.S. DeVore and S.F. Gull an C.K. Johnson. *Reconstruction of calmodulin single-molecule FRET states, dye interactions, and CaMKII peptide binding by MultiNest and classic maximum entropy*. 2012 (cit. on p. 1).

[7]R. Enberg, S. Munir, C. Pérez de los Heros, and D. Werder. *Prospects for higgsino-singlino dark matter detection at IceCube and PINGU*. http://arxiv.org/pdf/1506.05714v3.pdf. 2015 (cit. on p. 1).

[8]F. Feroz and M.P. Hobson. *Multimodal nested sampling: an efficient and robust alternative to MCMC methods for astronomical data analysis*. http://arxiv.org/pdf/0704.3704.pdf. 2007 (cit. on pp. 4, 5, 7, 21).

[9]F. Feroz, K. Cranmer, M. Hobson, R. Ruiz de Austri, and Roberto Trotta. *Challenges of Profile Likelihood Evaluation in Multi–Dimensional SUSY Scans*. http://arxiv.org/pdf/1101.3296v2.pdf. 2011 (cit. on pp. 5, 21).

[10]F. Feroz, J.R. Gair, P. Graff, M.P. Hobson, and A. Lasenby. *Classifying LISA gravitational wave burst signals using Bayesian evidence*. 2010 (cit. on p. 1).

[11]F. Feroz, M.P. Hobson, E. Cameron, and A.N. Pettitt. *Importance Nested Sampling and the MultiNest Algorithm*. http://arxiv.org/pdf/1306.2144.pdf. 2014 (cit. on pp. 15, 21).

[12]F. Feroz, M.P. Hobson, and M. Bridges. *MULTINEST: an efficient and robust Bayesian inference tool for cosmology and particle physics*. http://arxiv.org/pdf/0809.3437v1.pdf. 2008 (cit. on pp. 5, 6, 8, 21, 24, 26, 43, 44, 46).

[13]F. Feroz, J.R. Gair, M.P. Hobson, and E.K. Porter. *Use of the MULTINEST algorithm for gravitational wave data analysis*. 2009 (cit. on p. 1).

[14]Farhan Feroz. *Code repository for MultiNest* (cit. on pp. 8, 13, 15, 29).

[15]Farhan Feroz and John Skilling. *Exploring Multi-Modal Distributions with Nested Sampling*. http://arxiv.org/pdf/1312.5638.pdf. 2013 (cit. on p. 21).

[16]P. Graff, F. Feroz, M.P. Hobson, and A. Lasenby. *BAMBI: blind accelerated multimodal Bayesian inference*. http://mnras.oxfordjournals.org/content/421/1/169.full.pdf+html. 2011 (cit. on p. 1).

[17]Patrick Hallen. *On the Measurement of High-Energy Tau Neutrinos with IceCube*. RWTH Aachen University, 2013, pp. 39–46 (cit. on p. 1).

[18]E.T. Jaynes. *Probability Theory - The Logic of Science*. Cambridge University Press, 2003, pp. 172–177 (cit. on p. 7).

[19]B.H.H. Juurlink and C.H. Meenderinck. *Amdahl's Law for Predicting the Future of Multicores Considered Harmful*. Vol. 40, No. 2. http://www.redaktion.tu-berlin.de/fileadmin/fg196/publication/amdahl2012.pdf. ACM SIGARCH Computer Architecture News, 2012 (cit. on p. 13).

[20]N.V. Karpenka, M.C. March, F. Feroz, and M.P. Hobson. *Bayesian constraints on dark matter halo properties using gravitationally lensed supernovae*. http://arxiv.org/pdf/1207.3708v2.pdf. 2013 (cit. on p. 1).

[21]R.E. Kass and A.E. Raftery. *Bayes factors*. Journal of the American Statistical Association 90, 1995, pp. 773–795 (cit. on p. 4).

[22]M.O. Khan, G. Kennedy, and D.Y.C. Chan. *A scalable parallel Monte Carlo method for free energy simulations of molecular systems*. 2004 (cit. on p. 13).

[23]Justin Lanfranchi. *A Quality-of-Fit Indicator for Reconstructions of Neutrino Interactions in the IceCube-PINGU Detector*. Vol. 59, Number 9. Annual Meeting of the Mid-Atlantic Section of the APS, 2014 (cit. on p. 1).

[24]Lin Lu, Yi-Kin Choi, Wenping Wang, and Myung-Soo Kim. *Variational 3D Shape Segmentation for Bounding Volume Computation*. Vol. 26, Issue 3. Computer Graphics Forum, 2007, p. 239 (cit. on p. 9).

[25]Sudhanshu Mishra. *Some new test functions for global optimization and performance of repulsive particle swarm method*. https://mpra.ub.uni-muenchen.de/2718/1/MPRA_paper_2718.pdf. North-Eastern Hill University, Shillong (India), 2006 (cit. on p. 21).

[26]Pia Mukherjee, David Parkinson, and Andrew R. Liddle. *A nested sampling algorithm for cosmological model selection*. http://arxiv.org/pdf/astro-ph/0508461v2.pdf. 2006 (cit. on pp. 8, 17).

[27]W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery. *Numerical Recipes in C - The Art of Scientific Computing*. Vol. 2. Cambridge University Press, 1992, pp. 131–136 (cit. on p. 6).

[28]Jeffrey S. Rosenthal. *Parallel computing and Monte Carlo algorithms*. 2000 (cit. on p. 13).

[29]J.R. Shaw, M. Bridges, and M.P. Hobson. *Efficient Bayesian inference for multimodal problems in cosmology*. http://lanl.arxiv.org/pdf/astro-ph/0701867v2. 2007 (cit. on p. 8).

[30]John Skilling. *Bayesian Analysis - Nested Sampling for General Bayesian Computation*. Vol. 1, Number 4. 2006, pp. 833–860 (cit. on pp. 4, 6, 7, 21).

[31]Wolfgang Tschirk. *Statistik: Klassisch oder Bayes - Zwei Wege im Vergleich*. German. Springer Spektrum, 2014 (cit. on p. 3).

[32]Martin Wolf. *Indirect Searches for Galactic Dark Matter with IceCube-DeepCore and PINGU*. `http://www.diva-portal.org/smash/get/diva2:726605/FULLTEXT01.pdf`. Stockholm University, 2014 (cit. on p. 1).

List of Figures

List of Tables

List of Algorithms

Declaration

I hereby declare that I have written the present thesis independently and without use of other than the indicated means. I also declare that to the best of my knowledge all passages taken from published and unpublished sources have been referenced. The paper has not been submitted for evaluation to any other examining authority nor has it been published in any form whatsoever.

Mainz, January 26, 2016

Maicon Hieronymus