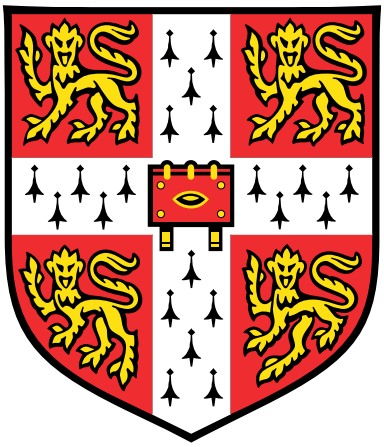
Bayesian Methods and Machine Learning in Astrophysics

Edward John Higson
Cavendish Astrophysics Group
Gonville & Caius College

1st October 2018



A dissertation submitted for the degree of Doctor of Philosophy at the University of Cambridge

Declaration

This dissertation is the result of my own work and includes nothing which is the outcome of work done in collaboration except as declared in the Preface and specified in the text.

The use of "we" is a stylistic choice.

It is not substantially the same as any that I have submitted, or, is being concurrently submitted for a degree or diploma or other qualification at the University of Cambridge or any other University or similar institution except as declared in the Preface and specified in the text. I further state that no substantial part of my dissertation has already been submitted, or, is being concurrently submitted for any such degree, diploma or other qualification at the University of Cambridge or any other University or similar institution except as declared in the Preface and specified in the text.

It does not exceed the prescribed word limit for the relevant Degree Committee (60,000 words).

Much of the material in this thesis has also been presented in Higson et al. (2018, 2019a,b,c); this work was done in collaboration with the candidate's supervisors (Anthony Lasenby, Mike Hobson and Will Handley).

Acknowledgements

I am extremely grateful to Anthony Lasenby and Mike Hobson for all their help and guidance over the last 3 years. I could not have wished for better supervisors; it has been a privilege working with them and I have learned an immense amount from the experience. I am equally grateful to Will Handley for his triple role in my PhD — as a third supervisor, a perennial code fixer and a friend.

I would also like to take this opportunity to thank the many people who inspired and fed my interest in physics and research over my many years in education. This includes (but is by no means limited to) David Wolfe and Anton Machacek at the Royal Grammar School High Wycombe, Peter Norreys and Raoul Trines at the Rutherford Appleton Laboratory, and Tony Weidberg, Georg Viehhauser, Devinder Sivia, Andrei Starinets and Ralph Schönrich at Oxford University.

I am grateful to my officemates Bjoern Soergel, Carina Negreanu, Iulia Simion, Fruzsina Agocs, Lukas Hergt and Pablo Lemos-Portela for their help with (and distraction from) various research challenges. Special thanks also goes to my fellow Cambridge-based expatriates from The Other Place — Jack Anthony, Jacob Swain and Peter Taylor — for their proof-reading services.

Most importantly I thank my loving parents, Jackie and Mark, for all the unwavering support (financial, emotional and logistical) they have provided throughout my life.

I dedicate this thesis to Harriet Smith, who has put up with me for 9 years now and I hope will continue to do so.

Summary

This thesis is concerned with methods for Bayesian inference and their applications in astrophysics. We principally discuss two related themes: advances in nested sampling (Chapters 3 to 5), and Bayesian sparse reconstruction of signals from noisy data (Chapters 6 and 7).

Nested sampling is a popular method for Bayesian computation which is widely used in astrophysics. Following the introduction and background material in Chapters 1 and 2, Chapter 3 analyses the sampling errors in nested sampling parameter estimation and presents a method for estimating them numerically for a single nested sampling calculation (this was published in Higson et al., 2018). Chapter 4 introduces diagnostic tests for detecting when software has not performed the nested sampling algorithm accurately, for example due to missing a mode in a multimodal posterior, and uses material from Higson et al. (2019b). The uncertainty estimates and diagnostics in Chapters 3 and 4 are implemented in the `nestcheck` (Higson, 2018b) software package.

Chapter 5, presented in Higson et al. (2019a), describes dynamic nested sampling: a generalisation of the nested sampling algorithm which can produce large improvements in computational efficiency compared to standard nested sampling. We have implemented dynamic nested sampling in the `dyPolyChord` (Higson, 2018a) and `perfectns` (Higson, 2018c) software packages.

Chapter 6 presents a principled Bayesian framework for signal reconstruction, in which the signal is modelled by basis functions whose number (and form, if required) is determined by the data themselves. This approach is based on a Bayesian interpretation of conventional sparse reconstruction and regularisation techniques, in which sparsity is imposed through priors via Bayesian model selection. We demonstrate our method for noisy 1- and 2-dimensional signals, including examples of processing astronomical images. The numerical implementation uses dynamic nested sampling, and uncertainties

are calculated using the methods introduced in Chapters 3 and 4. Chapter 7 applies our Bayesian sparse reconstruction framework to artificial neural networks, where it allows the optimum network architecture to be determined by treating the number of nodes and hidden layers as parameters. Chapters 6 and 7 use material from Higson et al. (2019c).

We conclude by suggesting possible areas of future research in Chapter 8.

Contents

Chapter 1

Introduction

As astrophysicists, we aim to create theories which describe the natural phenomena we observe in the universe. The scientific method requires that we test our hypotheses empirically; this allows them to be falsified or refined, and enables us to choose between rival models. Consequently, making inferences from data is fundamental to the research process.

This empirical scientific method is not new; a detailed account can be found in Sir Francis Bacon's 1620 work *Novum Organum Scientiarum* ("New Instrument of Science"). However, modern advances in experiments and computing have made data analysis techniques more important to scientific research now than ever before. In astrophysics in particular, the last two decades have seen order-of-magnitude increases in the quantity of both data and computational resources available. This has revolutionised our understanding of many aspects of the universe, and made advances in data analysis techniques and numerical methods central to the progress of the field.

The first theme of this thesis, *advances in nested sampling*, contributes to solving the challenges posed by these recent increases in the size and complexity of astronomical data. After a review of Bayesian inference methods and theory in Chapter 2, Chapters 3 and 4 provide practical tools for assessing the uncertainty and reliability of nested sampling calculations — which are poorly understood compared to many other numerical methods. Such calculations underpin a large number of recent results in astrophysics, but their reliability is not guaranteed; properly checking results is therefore of great scientific importance. Astronomical applications of the techniques introduced can be found in Sections 3.7 and 4.7.

Chapter 5 introduces dynamic nested sampling: a new algorithm which can provide order-of-magnitude improvements in computational efficiency over standard nested sampling, permitting the analysis of larger and more complex data sets. Since the publication of the dynamic nested sampling algorithm, its applications in astronomy have included constraining the present day stellar mass function (Orazio et al., 2018), fitting light curves of transient sources (Guillochon et al., 2018) and mapping distances across the Perseus molecular cloud (Zucker et al., 2018).

The thesis' second theme, *Bayesian sparse reconstruction of signals*, introduces a principled approach for fitting data using machine learning techniques such as regression[1] and neural networks. In Chapter 6 we describe our Bayesian sparse reconstruction framework, in which signals are fitted using basis functions whose number (and, if required, form) is determined by the data themselves. We show that this can be done by treating the type and number of basis functions as integer parameters, then performing Bayesian model selection indirectly by sampling the posterior distribution of these parameters. Chapter 7 applies this approach to artificial neural networks, where it allows Bayesian model selection to be performed over the space of possible network architectures by treating the number of hidden layers and nodes as integer parameters. Demonstrations of our method in Chapters 6 and 7 include applications to processing astronomical images.

Our Bayesian sparse reconstruction research is closely linked to recent advances in computing power and numerical methods such as nested sampling, which have made this principled approach feasible in the low data regime. We also make use of diagnostic tests and the dynamic nested sampling algorithm, introduced in Chapters 3 to 5, in the numerical calculations in Chapters 6 and 7. We intend the examples of Bayesian sparse reconstruction in this thesis to serve as a proof of principle for a wider range of astronomical applications, which will be made possible by future improvements in computational hardware and numerical methods.

[1]The modern field of "machine learning" incorporates a number of classical statistical techniques which significantly predate the coining of the term by Arthur Samuel in the 1950s. This includes regression, the earliest form of which dates back to Legendre (1805).

Chapter 2

Bayesian inference

We now provide an overview of Bayesian inference, which is central to the work in subsequent chapters of this thesis. Many excellent books on this topic exist; in particular we use Sivia and Skilling (2006) and MacKay (2003). In addition there are some good review articles on Bayesian methods in the context of astrophysics and cosmology; we have drawn from Trotta (2008), Loredo (2012) and Sharma (2017). The chapter finishes with an introduction to nested sampling in Section 2.4, which will provide background for Chapters 3 to 5. Some sections of this chapter have been adapted from background material presented in Higson et al. (2018, 2019c).

2.1 Bayesians and frequentists

We first introduce Bayesian inference by contrasting it with the rival frequentist paradigm.

2.1.1 Probability

The distinction between Bayesians and frequentists can be understood in terms of the two schools' differing definitions of probability. A frequentist definition is:

> "Probability is an event's relative frequency in the limit of an infinite number of independent trials."

At first glance this definition makes intuitive sense, but it has a number of significant shortcomings identified by Trotta (2008). These include:

1. it cannot formulate perfectly reasonable questions involving unrepeatable events. For example, "what is the probability that King Richard III arranged the murder of his two young nephews (the princes in the tower) in 1483?";

2. to hold exactly it requires an infinite number of trials, which are never available in practice. Small sample sizes require *ad hoc* and potentially complex adjustments;

3. it is circular, in that it assumes the event has the same probability of occurring in each trial when this probability is exactly what we seek to define;

4. it covers only random processes (aleatoric uncertainty), such as the probability of an atomic nucleus undergoing radioactive decay within some time period. Epistemic uncertainty due to a lack of knowledge about a deterministic system is not included.

As a consequence, this definition of "probability" is often radically different to the word's meaning when it is used colloquially. Imagine you suggest to your friend that you leave a cake in the oven for 45 minutes to bake, and they reply "if we do that, there is a 50% probability we will burn it!" Does your friend mean to tell you that, if you left a very large number of identical cakes in identical ovens at identical temperatures for 45 minutes, approximately half of them would be burned and half of them not burned? This seems somewhat unlikely. More plausibly, rather than suggesting the baking process is stochastic, your friend believes it is broadly deterministic but is unsure of the outcome.

In such a conversation we implicitly assume a Bayesian definition of probability, which does not suffer from the problems with the frequentist version identified above:

> "Probability is a measure of the degree of belief that a proposition is true."

This definition is not circular, and can clearly be applied to unrepeatable events and limited numbers of samples. Furthermore, it does not distinguish between uncertainty due to a process' intrinsic randomness and due to lack of information.

2.1.2 Bayes' theorem

In order to use the Bayesian definition of probability to analyse new data, we require a method for updating our degree of belief in a proposition given new information. The mathematical formula for this procedure is credited to the Reverend Thomas Bayes

(1701(?)-1761), from whom Bayesian statistics takes its name. Bayes' eponymous theorem was published posthumously in 1763, by his friend the philosopher Richard Price (Bayes and Price, 1763).

Bayes' theorem can be derived from the axioms required for consistent reasoning (the "Cox axioms"), which imply that the probabilities of propositions X and Y must satisfy the sum and product rules (Sivia and Skilling, 2006). The sum rule states

$$P(X|I) + P(\bar{X}|I) = 1, \tag{2.1}$$

where $\bar{X}$ denotes "not X" and $P(X|I) \in [0, 1]$. Here $P(X|I)$ denotes the conditional probability of X given I, and following Sivia and Skilling (2006) we explicitly include that all probabilities are conditional on any background information I. Such conditional probabilities represent logical rather than causal or temporal connections; for example future events can provide us with more information about the probabilities of past events. The product rule is

$$P(X, Y|I) = P(X|Y, I)P(Y|I). \tag{2.2}$$

Bayes' theorem can be easily derived from (2.2) by noting $P(X, Y|I) = P(Y, X|I)$, so

$$P(X|Y, I)P(Y|I) = P(Y|X, I)P(X|I), \tag{2.3}$$

and hence

$$P(X|Y, I) = \frac{P(Y|X, I)P(X|I)}{P(Y|I)}. \tag{2.4}$$

This provides a formula for updating our prior degree of belief $P(X|I)$, termed the *prior*, with new information Y. It is worth mentioning that Bayes' theorem is an uncontroversial mathematical statement, and the disagreement between Bayesians and frequentists is about its use as a basis for inference (Trotta, 2008).

Until this point we have focused on "propositions" X and Y which can be true or false, but Bayes' theorem (2.4) and the Bayesian definition of probability also extend to quantities which can take a range of discrete or continuous values. In the discrete case X takes values in $[X_1, X_2, \dots]$, and the sum rule (2.1) becomes

$$\sum_i P(X_i|I) = 1, \tag{2.5}$$

where $P(X_i|I) \in [0, 1]$ for all i. In the continuous case X takes values in $\mathbb{R}$ and

$$\int_{-\infty}^{\infty} P(X|I)\,\mathrm{d}X = 1, \tag{2.6}$$

where $P(X|I) > 0$ for all $X \in \mathbb{R}$. Another significant result which follows from the Cox axioms is "marginalisation" (also called the "law of total probability"), which for continuous variables X and Y can be written as

$$P(X|I) = \int_{-\infty}^{\infty} P(X, Y|I)\,\mathrm{d}Y. \tag{2.7}$$

This allows the removal ("marginalising out") of parameters from joint distributions, and is of great importance for Bayesian inference. We use the notation in this section for the remainder of this thesis, although more rigorous conventions are common in the statistics literature.

2.1.3 Practical differences

Perhaps the most commonly cited difference between the Bayesian and frequentist approaches is the former's requirement that the initial knowledge of the proposition is specified mathematically in the prior. This introduces additional subjectivity and complexity, and is often viewed by non-Bayesians as a disadvantage. Indeed simply specifying an absence of knowledge — a non-informative prior — often requires careful analysis (for some examples see Handley and Millea, 2018; Sivia and Skilling, 2006). However it can be argued that the inclusion of the prior is not a limitation but a feature of the Bayesian approach (Trotta, 2008), as it serves to make assumptions explicit and to model the fact that different scientists can interpret the same data differently given their distinct previous experiences. In addition there are many circumstances when there is an uncontroversial basis for the inclusion of prior information, such as when analysing noisy measurements of a physical variable that we know must be positive or take a value within some range.

We take the view of Loredo (2012), who argues the most fundamental difference between calculations using the two approaches is not the modulation by the prior but the space over which the analysis takes place. Whereas a frequentist calculation takes place in the sample space (the space of possible measurements), Bayesian computation is performed in the parameter (hypothesis) space. This perspective elucidates the necessity

of the prior to provide a measure over the parameter space to be analysed. If the prior were not included, inferences about some physical quantity involving averaging it (integrating its distribution over the parameter space) could be affected by simple reparameterisations; for an interesting discussion of this see Loredo (2012, p10-12). The exploration of the parameter space is a key aspect of the Bayesian computation techniques discussed in Section 2.3.

2.1.4 Why isn't everyone a Bayesian?

Given the philosophical advantages of the Bayesian approach discussed in Section 2.1.1 and the persuasive arguments presented by Trotta (2008), Sivia and Skilling (2006) and Loredo (2012) among others, it is reasonable to ask why everyone is not a Bayesian. In fact, historically, most 19^{th} century scientists used a Bayesian perspective, including notably Laplace — who derived Bayes' theorem independently and first expressed it in its modern form (2.4) (Sivia and Skilling, 2006).

This changed in the 20^{th} century with the introduction of two successful rival frequentist frameworks: Fisher's "significance testing" and Neyman-Pearson "hypothesis testing". These approaches introduce a variety of procedures without a clear overarching rationale (Sivia and Skilling, 2006). However they claim the advantages of ease of application and objectivity, in particular in comparison to the careful thought[1] often needed to choose the priors in a Bayesian analysis; interesting discussions can be found in Efron (1986) and Gelman (2008).

Unsurprisingly many Bayesians dispute these claims, including Loredo (2012) and Sivia and Skilling (2006). Furthermore frequentist statistics can also lead to subtle problems when not used carefully, with p-value significance tests believed to be a major cause of the "replication crisis" currently being experienced in the social sciences (for more details see the recent American Statistical Association statement by Wasserstein and Lazar, 2016). We are currently seeing rapid growth in the popularity of Bayesian methods, in particular in astrophysics, and many predict the 21^{st} century will see a return to the dominance of the Bayesian school (Loredo, 2012; Lindley, 1975). However, the debate is far from settled (Trotta, 2008).

[1] Gelman (2008), writing in the voice of a hypothetical anti-Bayesian, quips that "recommending that scientists use Bayes' theorem is like giving the neighborhood kids the key to your F-16."

Having provided some context, the remainder of this chapter and this thesis will now focus on the Bayesian approach.

2.2 Applying Bayes' theorem to data

Scientific research is about creating models to explain and understand available data. Models and their parameters offer a description of the universe, which we use to work out how likely we are to observe a given set of data values. However, as scientists our primary goal is to solve the inverse problem — i.e. make an inference about the state of the universe (which model is correct, and what are the model's parameter values) given the data. Such inferences can be divided into *parameter estimation* and *model selection*.

Given some model $\mathcal{M}$, parameter estimation involves determining the values of its parameters $\boldsymbol{\theta}$ using the data $\mathcal{D}$. Bayes' theorem (2.4) can be applied to parameter estimation by replacing I with the model $\mathcal{M}$, X with the model's parameters $\boldsymbol{\theta}$ and Y with the data $\mathcal{D}$. This gives

$$P(\boldsymbol{\theta}|\mathcal{D},\mathcal{M}) = \frac{P(\mathcal{D}|\boldsymbol{\theta},\mathcal{M})P(\boldsymbol{\theta}|\mathcal{M})}{P(\mathcal{D}|\mathcal{M})}, \tag{2.8}$$

which we write schematically as

$$\mathcal{P}(\boldsymbol{\theta}) = \frac{\mathcal{L}(\boldsymbol{\theta})\pi(\boldsymbol{\theta})}{\mathcal{Z}}. \tag{2.9}$$

Here the *prior*

$$\pi(\boldsymbol{\theta}) \equiv P(\boldsymbol{\theta}|\mathcal{M}) \tag{2.10}$$

represents our knowledge of the model's parameters in the absence of the data. The model tells the probability of observing data $\mathcal{D}$ given some set of parameter values $\boldsymbol{\theta}$, and can therefore be expressed as a distribution $P(\mathcal{D}|\boldsymbol{\theta},\mathcal{M})$ which we term the *likelihood* and write as

$$\mathcal{L}(\boldsymbol{\theta}) \equiv P(\mathcal{D}|\boldsymbol{\theta},\mathcal{M}). \tag{2.11}$$

Thus Bayes' theorem allows us to update our prior knowledge $\pi(\boldsymbol{\theta})$ given new information $\mathcal{D}$ to obtain the *posterior distribution*

$$\mathcal{P}(\boldsymbol{\theta}) \equiv P(\boldsymbol{\theta}|\mathcal{D},\mathcal{M}). \tag{2.12}$$

$\lvert \log \mathcal{B}^j_k \rvert$	Odds	Notes
< 1.0	$\lessapprox 3:1$	Inconclusive
1.0	$\approx 3:1$	Positive evidence
2.5	$\approx 12:1$	Moderate evidence
5.0	$\approx 150:1$	Strong evidence

Table 2.1: The "Jeffreys' scale" for interpreting the Bayes factor between two models $\mathcal{M}_j$ and $\mathcal{M}_k$, reproduced from Trotta (2007). The first column shows the log Bayes factor value, the second column shows the approximate relative odds of the two models being correct and the third column gives a qualitative interpretation.

The *Bayesian evidence* $\mathcal{Z}$ is a normalisation constant, and is computed by averaging the likelihood $\mathcal{L}(\boldsymbol{\theta})$ over the prior $\pi(\boldsymbol{\theta})$

$$\mathcal{Z} \equiv P(\mathcal{D}|\mathcal{M}) = \int \mathcal{L}(\boldsymbol{\theta})\pi(\boldsymbol{\theta})\,\mathrm{d}\boldsymbol{\theta}. \tag{2.13}$$

Bayes' theorem can also be used to compare different models $\mathcal{M}_1, \mathcal{M}_2, \ldots$ and assess which best describes the data. The posterior probability of a given model is

$$P(\mathcal{M}_j|\mathcal{D}) = \frac{P(\mathcal{D}|\mathcal{M}_j)P(\mathcal{M}_j)}{P(\mathcal{D})} = \frac{\mathcal{Z}_j\Pi_j}{\sum_k \mathcal{Z}_k\Pi_k}, \tag{2.14}$$

where $\Pi_j \equiv P(\mathcal{M}_j)$ denotes the prior probability of each model and the denominator of the final term sums over all competing models. The evidence $\mathcal{Z}$ penalises more complex models so this approach naturally includes Occam's razor. Models may also be compared by computing log posterior odds ratios

$$\mathcal{P}^j_k \equiv \log\left(\frac{P(\mathcal{M}_j|\mathcal{D})}{P(\mathcal{M}_k|\mathcal{D})}\right) = \log\left(\frac{\mathcal{Z}_j}{\mathcal{Z}_k}\right) + \log\left(\frac{\Pi_j}{\Pi_k}\right), \tag{2.15}$$

where here and in the remainder of this thesis log denotes the natural logarithm. The ratio of evidences $\mathcal{B}^j_k = \mathcal{Z}_j/\mathcal{Z}_k$ is called a Bayes factor; Bayes factors are independent of the prior Π_j on different models $\mathcal{M}_j$, but depend on the priors on the models' parameters $\pi(\boldsymbol{\theta}_{\mathcal{M}_j})$ through the calculation of $\mathcal{Z}_j$ from (2.13). If the prior Π_j on different models is uniform, the Bayes factors are equal to the posterior odds ratios. The "Jeffreys' scale" (Jeffreys, 1961) provides a numerically calibrated scale for qualitatively interpreting Bayes factors; we reproduce a modified version from Trotta (2007) in Table 2.1.

2.3 Bayesian computation

In some simple cases, parameter estimation (2.9) and model comparison (2.14) calculations can be performed analytically. However for most problems in astronomy this is not possible, and numerical calculations are required. One technique for doing this is *nested sampling*, which is a major focus of this thesis. This section provides an overview of some Bayesian computation techniques which are popular in astrophysics, in order to provide context for our discussion of nested sampling in Section 2.4. Many excellent guides to this topic are available; we have used Sharma (2017), Hogg and Foreman-Mackey (2018) and Feroz (2008).

Likelihoods $\mathcal{L}(\boldsymbol{\theta})$ are often computationally expensive functions, so the goal of Bayesian computation is to obtain posterior inferences using a limited number of evaluations of the likelihood function ("likelihood calls"). The number of likelihood calls required for numerical integration (for example by quadrature) increases exponentially with the dimensionality of the parameter space, so this approach is typically impractical except in very low dimensions. As a result Bayesian computation is usually carried out using Monte Carlo methods, which involve repeated random sampling.

2.3.1 Parameter estimation

Parameter estimation calculations can be performed by generating a set of samples from the posterior distribution, then using these to make inferences about quantities of interest such as the posterior means of parameters. Samples can also be used to numerically estimate the posterior distributions of parameters or functions of parameters with kernel density estimation.

In astronomy the most popular approach for generating samples is to use Markov chain Monte Carlo (MCMC); a class of methods for sampling probability distributions which explore the parameter space via a biased random walk (Hogg and Foreman-Mackey, 2018). Samples produced by these methods form a Markov chain, meaning that the probability distribution of the next random variable $\boldsymbol{\theta}_{i+1}$ depends only on the current state $\boldsymbol{\theta}_i$ and is independent of the previous evolution of the sequence. The chains should have the property that, after a large number of steps from the starting point, the samples produced will have an invariant limiting distribution which is proportional to the posterior distribution (hence the chains must be *ergodic*). This can be achieved if

the rule for selecting a new point satisfies certain conditions (see Sharma, 2017; MacKay, 2003, for a detailed discussion). Most MCMC methods satisfy *detailed balance*, meaning that the probability of being in one state and transitioning to another state is the same in either direction and as a result the process is reversible.

The Metropolis-Hastings algorithm (Metropolis et al., 1953; Hastings, 1970) is the most general MCMC algorithm (Sharma, 2017), and is shown in Algorithm 1. New points are added via a two step process: first a candidate point is sampled from a proposal distribution $q(\boldsymbol{\theta}'|\boldsymbol{\theta}_i)$, then it is accepted or rejected with a probability which depends on the value of the posterior at the candidate point $\mathcal{P}(\boldsymbol{\theta}')$ relative to the posterior at the previous point $\mathcal{P}(\boldsymbol{\theta}_i)$. Most variants use symmetric proposal distributions which satisfy $q(\boldsymbol{\theta}'|\boldsymbol{\theta}_i) = q(\boldsymbol{\theta}_i|\boldsymbol{\theta}')$; this simplifies the condition for acceptance to $U < \mathcal{P}(\boldsymbol{\theta}')/\mathcal{P}(\boldsymbol{\theta}_i)$ (where $U \in [0, 1]$ is a uniform random variable). In this case, if $\mathcal{P}(\boldsymbol{\theta}') > \mathcal{P}(\boldsymbol{\theta}_i)$ then the candidate point is guaranteed to be accepted — but it may be rejected otherwise. This biases the random walk towards regions where $\mathcal{P}(\boldsymbol{\theta})$ is high, and is the mechanism by which the distribution of samples produced is made proportional to $\mathcal{P}(\boldsymbol{\theta})$.

Result: Samples $\{\boldsymbol{\theta}_i\}$ from the posterior $\mathcal{P}(\boldsymbol{\theta})$.
Input: Posterior distribution function $\mathcal{P}(\boldsymbol{\theta})$, starting point $\boldsymbol{\theta}_1$, proposal distribution $q(\boldsymbol{\theta}'|\boldsymbol{\theta}_i)$.
for $i = 1$ **to** $N - 1$ **do**
  sample $\boldsymbol{\theta}'$ from $q(\boldsymbol{\theta}'|\boldsymbol{\theta}_i)$;
  sample uniform random variable $U \in [0, 1]$;
  if $U < \frac{\mathcal{P}(\boldsymbol{\theta}')q(\boldsymbol{\theta}_i|\boldsymbol{\theta}')}{\mathcal{P}(\boldsymbol{\theta}_i)q(\boldsymbol{\theta}'|\boldsymbol{\theta}_i)}$ **then**
    $\boldsymbol{\theta}_{i+1} = \boldsymbol{\theta}'$;
  else
    $\boldsymbol{\theta}_{i+1} = \boldsymbol{\theta}_i$;
  end
end

Algorithm 1: The Metropolis-Hastings algorithm.

There are many variants of the Metropolis-Hastings approach (Algorithm 1). These include:

Gibbs sampling: different parameters (components of $\boldsymbol{\theta}$) are updated individually using sampling from conditional distributions (Geman and Geman, 1984). Unlike in Algorithm 1, all samples are accepted.

Adaptive MCMC: the proposal distribution is dynamically adapted based on past samples (see Andrieu and Thoms, 2008, for a review).

Affine invariant ensemble samplers: an ensemble of interacting Markov chains is used, and the resulting process' performance is unaffected by affine transformations of the parameter space (Goodman and Weare, 2010). The popular `emcee` package (Foreman-Mackey et al., 2013) uses this approach.

Hamiltonian Monte Carlo methods: an auxiliary momentum variable is added for each parameter, and Hamiltonian dynamics are used to assist in the sampling of new points (for a recent review see Betancourt, 2017). A version by Hoffman and Gelman (2014), in which the sizes of steps between points are set adaptively, is used in the popular Bayesian inference package `stan`.

Parallel tempering: uses an ensemble of samplers which can exchange information and target different powers of the distribution ("temperatures") $\mathcal{P}(\boldsymbol{\theta})^{1/T_i}$. The lowest temperature $T_n = 1$ samples the posterior and other temperatures $T_{i \neq n} > 1$ broaden the target distribution to allow a wider exploration of the parameter space (Earl and Deem, 2005).

Limitations

The Metropolis-Hastings algorithm has some significant practical difficulties and drawbacks which are largely shared by similar MCMC approaches. The algorithm requires a proposal distribution must be specified, which greatly affects the efficiency of the process and can be challenging to choose *a priori*. Furthermore, successive samples in the chain are correlated, and "convergence diagnostics" are required to work out how long the process must be run for in order that the samples give a good approximation to the posterior (for a detailed guide see Cowles and Carlin, 1996). Similarly the first portion of the samples, which are correlated with the starting point, must be removed (this is referred to as "burn in").

However the most significant limitation of the Metropolis-Hastings algorithm is its inefficiency when exploring multimodal posteriors; it can take an intractably large number of iterations for a chain to transition between two modes separated by a wide region of low $\mathcal{P}(\boldsymbol{\theta})$ values. Similar problems occur with curving degeneracies, as the small step

length required to avoid stepping off the maxima in the directions in which it is thin means exploration along its length takes a very large number of steps (Feroz, 2008). Multimodal distributions are problematic for all MCMC approaches; they can be partially addressed with parallel tempering (see above), but this adds computational cost and complexity (Sharma, 2017).

Another major limitation of the Metropolis-Hastings algorithm is its inability to effectively compute the Bayesian evidence (2.13), which is used in Bayesian model selection. We now discuss numerical methods for Bayesian model selection.

2.3.2 Model selection

Calculation of the Bayesian evidence (2.13) is computationally challenging since it involves a (possibly high-dimensional) integral over the parameter space. In principle an estimate of the integral's value can be found from posterior samples produced by the Metropolis-Hastings algorithm and similar MCMC methods, but in practice this is highly computationally inefficient. A major reason for this is that MCMC focuses on sampling the posterior's peak, leading to inaccuracies in the integral due to insufficient samples in the tails of the distribution (Feroz, 2008).

In addition to nested sampling (which will be discussed in the next section), a number of methods can be used to estimate the evidence or calculate Bayes factors directly. These include:

Thermodynamic integration (simulated annealing): samples are taken from $\mathcal{L}(\boldsymbol{\theta})^{\beta}\pi(\boldsymbol{\theta})$, with the cooling temperature β raised from 0 (the prior) to 1 (the posterior). The evidence can then be calculated as $\log \mathcal{Z} = \int_0^1 \mathrm{E}[\log \mathcal{L}(\boldsymbol{\theta})]_\beta \, \mathrm{d}\beta$, where $\mathrm{E}[\log \mathcal{L}(\boldsymbol{\theta})]_\beta$ is the expectation of $\log \mathcal{L}(\boldsymbol{\theta})$ over the distribution $\mathcal{L}(\boldsymbol{\theta})^{\beta}\pi(\boldsymbol{\theta})$ (Gelman and Meng, 1998; Kirkpatrick et al., 1983). However this approach is often computationally expensive compared to nested sampling (Feroz, 2008), and can fail under certain circumstances due to "phase changes" (see Sivia and Skilling, 2006, Section 9.6.1 for more details).

Sequential Monte Carlo (SMC) samplers: a sequence of probability distributions is sampled by evolving a "cloud" of weighted random variables (Del Moral et al., 2006). This approach can be used to provide both posterior samples and an

estimate of the Bayesian evidence, and is related to nested sampling (see Salomone et al., 2018, for more details) but is less popular in astronomy.

Variational Inference: a proxy distribution with some free parameters is proposed and fitted to the posterior, typically by minimising the Kullback-Leibler divergence. The known properties of the proxy distribution can then be used to estimate the evidence (Blei et al., 2017).

Laplace's method: estimates the evidence under the assumption that the posterior is approximately Gaussian (Tierney and Kadane, 1986).

Savage-Dickey density ratio: allows the Bayes factors between two nested models to be computed, provided one model is contained in the other and the more complex model is equal to the contained model for some choice of parameters (Verdinelli and Wasserman, 1995).

MCMC-based methods: techniques exist which allow Bayes factors to be computed from MCMC chains. These include product space MCMC (Green, 1995), which allows the sampler to jump between models (subspaces).

For a more detailed review of evidence estimation techniques, see Friel and Wyse (2012).

Bayesian predictive methods provide an alternative approach to model comparison which does not use (2.14). This involves estimating how well the model will fit new data, while adjusting for the fact it has been "trained" on the current data, using theoretically justified information criteria. Examples include the "Akaike information criterion" (AIC) and the "widely applicable information criterion" (WAIC), which can be evaluated from posterior samples (Akaike, 1974; Watanabe, 2010). Both of these criteria have an expected value equal to the Kullback-Leibler divergence of the predicted posterior from the true posterior, and are equivalent to leave-one-out cross-validation (LOOCV) in the limit of large sample size (Sharma, 2017). They can be useful when the goal is to test the predictive performance of models on new data, or when choosing priors and computing Bayes factors is difficult.

2.4 Nested sampling

The remainder of this chapter focuses on nested sampling (Skilling, 2004, 2006); a Monte Carlo method which simultaneously computes Bayesian evidences (2.13) and samples from the posterior distribution (2.9). The early development of the nested sampling algorithm was focused on evidence calculation, which is computationally challenging (as discussed in the previous section). However, contemporary implementations such as `MultiNest` (Feroz and Hobson, 2008; Feroz et al., 2008, 2013) and `PolyChord` (Handley et al., 2015a,b) are now also extensively used for parameter estimation from posterior samples (see for example Planck Collaboration, 2016a). Nested sampling compares favourably to MCMC-based parameter estimation for degenerate, multi-modal likelihoods as it has no "thermal" transition probability and exponentially compresses the prior distribution to the posterior. Allison and Dunkley (2014) empirically tests nested sampling parameter estimation against MCMC-based alternatives, and recommends its use over Metropolis-Hastings sampling (Algorithm 1) in many cases.

The remainder of this chapter provides a description of the nested sampling algorithm, and discusses how it can be implemented. For theoretical treatments of nested sampling's convergence properties, see Keeton (2011), Skilling (2009), Walter (2017) and Evans (2007).

2.4.1 The nested sampling algorithm

Initially n points, termed *live points*, are sampled randomly from the prior. At each iteration i, the live point with the lowest likelihood $\mathcal{L}_i$ is removed and replaced by a new live point sampled from the prior subject to the constraint that it has a likelihood higher than $\mathcal{L}_i$. Iterating until some termination condition is met generates a list of discarded samples known as *dead points*, which are used to estimate the evidence and make posterior inferences.[2] We refer to the completed nested sampling process as a *run*.

To compute the evidence, the many-dimensional integral (2.13) is reduced to a one-dimensional integral in terms of the fractional prior volume within an iso-likelihood contour. We define the fraction of the prior $\pi(\boldsymbol{\theta})$ with likelihood $\mathcal{L}(\boldsymbol{\theta})$ greater than

[2]The remaining live points at termination can also be used if required, but termination conditions can be chosen such that this makes a negligible difference to calculation results.

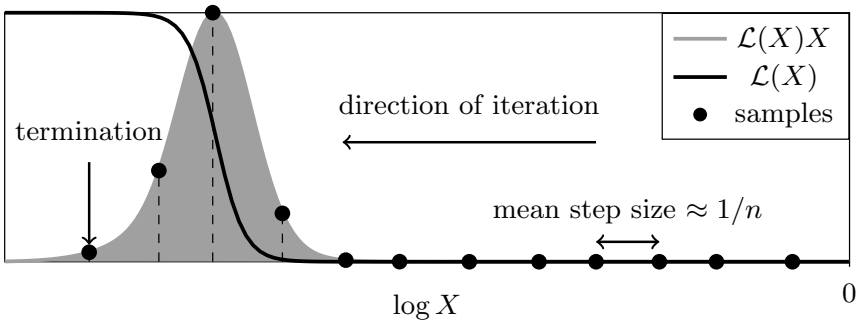


Figure 2.1: A schematic representation of nested sampling with a constant number of live points n. The curve $\mathcal{L}(X)X$ shows the relative posterior mass, the bulk of which is contained in some small fraction of the prior and is only visible on a log scale in X. The algorithm iterates inwards in X exponentially with stochastic shrinkage ratios distributed according to (2.18).

some value $\mathcal{L}^*$ as $X(\mathcal{L}^*)$, where

$$X(\mathcal{L}^*) \equiv \int_{\mathcal{L}(\boldsymbol{\theta})>\mathcal{L}^*} \pi(\boldsymbol{\theta})\,\mathrm{d}\boldsymbol{\theta}, \tag{2.16}$$

and $X \in [0,1]$. Provided the inverse $\mathcal{L}(X) \equiv X^{-1}(\mathcal{L})$ exists,[3] the evidence (2.13) can be expressed as

$$\mathcal{Z} = \int_0^1 \mathcal{L}(X)\,\mathrm{d}X. \tag{2.17}$$

Given a set of dead points with likelihoods $\mathcal{L}_i$, the corresponding prior volumes X_i are unknown but are modelled statistically as $X_i = t_i X_{i-1}$, where $X_0 = 1$ and each shrinkage ratio t_i is independently distributed as the largest of n random variables from the interval $[0,1]$ (Skilling, 2006). Hence:

$$P(t_i) = n t_i^{n-1}, \qquad \mathrm{E}[\log t_i] = -\frac{1}{n}, \qquad \mathrm{Var}[\log t_i] = \frac{1}{n^2}, \tag{2.18}$$

and the algorithm samples within an exponentially shrinking part of the prior. This exponential shrinkage is shown schematically in Figure 2.1.

[3] A sufficient condition for $\mathcal{L}(X) \equiv X^{-1}(\mathcal{L})$ to exist is for $\mathcal{L}$ to be continuous and π to have a connected support. See Chopin and Robert (2010) and Feroz et al. (2013, Appendix C) for a more detailed measure-theoretic discussion.

Evidence estimation

Nested sampling therefore allows one to approximate the evidence (2.17) via a quadrature sum over the dead points

$$\mathcal{Z}(\mathbf{t}) \approx \sum_{i \in \text{dead}} \mathcal{L}_i w_i(\mathbf{t}), \tag{2.19}$$

where $\mathbf{t} = \{t_1, t_2, \ldots, t_{n_{\text{dead}}}\}$ are the unknown set of shrinkage ratios for the n_{dead} iterations of the nested sampling process, and each t_i is an independent random variable drawn from distribution (2.18). The shrinkage ratios define the prior volumes via $X_i(\mathbf{t}) = \prod_{k=0}^{i} t_k$, and the w_i are appropriately chosen quadrature weights roughly corresponding to the volume of the "prior shell" to which a given dead point belongs. For example, using the trapezium rule: $w_i(\mathbf{t}) = \frac{1}{2}(X_{i-1}(\mathbf{t}) - X_{i+1}(\mathbf{t}))$.[4]

Given that the shrinkage ratios $\mathbf{t}$ are *a priori* unknown, we may quantify our knowledge of $\mathcal{Z}$ by simulating sets of $\mathbf{t}$ according to (2.18), and working with the distribution of the resulting set of evidences $\{\mathcal{Z}\}_{\mathbf{t}}$ from (2.19) (Skilling, 2006). Typically one then computes and reports a mean value and error for $\log \mathcal{Z}$ from this distribution.

Several alternative methods for calculating evidence inferences are reported in the literature. Skilling (2006) also proposes an error calculation based on relative entropy, which demonstrates that the uncertainty of $\log \mathcal{Z}$ is dominated by the Poisson variability in the number of steps required to reach the bulk of the posterior mass. Keeton (2011) uses distribution moments and running totals which are updated with each nested sampling step. This method has been extended by Handley et al. (2015b) to allow the splitting of multi-modal likelihoods into different clusters and the treatment of variable numbers of live points. For a more detailed discussion of the convergence properties of nested sampling evidences, see Chopin and Robert (2010).

Thus, the dominant sampling error in the evidence estimate (2.19) from perfect nested sampling is from statistical variation in the unknown volumes of the prior "shells" $w_i(\mathbf{t})$ that each point represents. The error from approximating the integral for $\mathcal{Z}$ with a sum can be safely neglected unless n is very small[5] (Skilling, 2006). There is also some

[4] For the final dead point, as $X_{n_{\text{dead}}+1}$ is not available, $w_{n_{\text{dead}}}(\mathbf{t})$ can be approximated with $w_{n_{\text{dead}}-1}(\mathbf{t})$. The remaining prior volume after the final dead point can be ignored, or included in calculations using the live points remaining at termination (a method for doing this is described in Section 5.2). The trapezium rule weight of the first point can be increased to assign it all prior volume between $X = 1$ and $X = X_1$, but this typically makes negligible difference in practice.

[5] The trapezium rule error is $\mathcal{O}(1/n^2)$, and if required other methods such as Simpson integration could be used.

error from terminating the algorithm and truncating the sum, but this is can be made negligible with appropriate termination conditions.

Parameter estimation

One may also perform posterior inference from nested sampling by using the dead points to construct a set of posterior samples with weights proportional to their share of the posterior mass (Skilling, 2006):

$$p_i(\mathbf{t}) = \frac{w_i(\mathbf{t})\mathcal{L}_i}{\sum_i w_i(\mathbf{t})\mathcal{L}_i} = \frac{w_i(\mathbf{t})\mathcal{L}_i}{\mathcal{Z}(\mathbf{t})}. \tag{2.20}$$

As before, $\mathbf{t}$ is the set of prior shrinkage ratios and in the trapezium rule case $w_i(\mathbf{t}) = \frac{1}{2}(X_{i-1}(\mathbf{t}) - X_{i+1}(\mathbf{t}))$. The resulting sampling errors are discussed in detail in Chapter 3.

2.4.2 Implementations

Significantly, the nested sampling algorithm as described above does not prescribe any specific method for generating samples from within iso-likelihood contours. Since nested sampling's inception, a variety of techniques have been applied to this problem. Some popular methods are (Handley et al., 2015b):

- MCMC-based sampling, the method originally envisaged by Skilling, which requires taking a number of steps along the MCMC chain between successive samples used in the nested sampling run to ensure they are decorrelated. However, traditional Metropolis-Hastings or Gibbs sampling requires a large amount of tuning of the proposal distribution to perform the required sampling efficiently (Feroz and Hobson, 2008). Two alternatives are Hamiltonian nested sampling (Betancourt, 2011) and Galilean nested sampling (Feroz and Skilling, 2013; Skilling, 2012), which use Hamiltonian Monte Carlo and Galilean Monte Carlo respectively. Both approaches have momentum-like auxiliary parameters and allow Markov chains to "bounce" off the hard likelihood constraint. However these approaches require gradients and a careful choice of step size, and can become inefficient for iso-likelihood contours which are difficult to bounce back into due to their shape (Feroz and Hobson, 2008).

- Rejection sampling is used in the popular `MultiNest` software (Feroz and Hobson, 2008; Feroz et al., 2008, 2013), which incorporates and improves an earlier

method introduced by Mukherjee et al. (2006). This involves enclosing the iso-likelihood contour in multiple overlapping ellipsoids, which are calculated using the live points, then sampling randomly from within this volume. Samples which do not satisfy the likelihood constraint are discarded. This is highly effective in modest dimensions, but the acceptance rate decreases exponentially with dimensionality.

- Slice sampling (Neal, 2003) was applied to nested sampling by Aitken and Akman (2013) and is used by `PolyChord` (Handley et al., 2015a,b). At each iteration, a random live point and a random direction are selected. Points are sampled along the line ("chord") with the chosen direction which intersects the chosen live point, in order to establish the bounds at which the line intersects the iso-likelihood contour. A point is then sampled along this line from within the contour. To prevent the new point from being correlated with the live point through which the chord passes, this process is repeated multiple times. While this is less efficient than rejection sampling in low dimensions, it becomes more efficient for high dimensional problems; computational costs with `PolyChord` scale as $\mathcal{O}(D^3)$ whereas for `MultiNest` they scale exponentially. For a more detailed description of `PolyChord`'s sampling method, including a diagram illustrating the process, see Handley et al. (2015b, Section 4).

Furthermore, a number of nested sampling variants have been proposed which alter the algorithm described in Section 2.4.1. Most notably, diffusive nested sampling (Brewer et al., 2011) uses a number of MCMC chains to sample a mixture of nested probability distributions. This approach is implemented in the `DNest` software package and has been applied to a number of astronomical problems (see for example Pancoast et al., 2014). In addition, superposition-enhanced nested sampling (Martiniani et al., 2014) combines nested sampling with global optimization techniques to improve the algorithm's ability to find all the modes in highly multimodal spaces. Salomone et al. (2018) propose another hybrid approach involving Sequential Monte Carlo (mentioned in Section 2.3).

Having provided an overview of Bayesian inference and nested sampling methods in this chapter, we next present our work on estimating the uncertainty and reliability of nested sampling calculations in Chapters 3 and 4. This is followed in Chapter 5 by a

description of our dynamic nested sampling algorithm, which can greatly increase the computational efficiency of nested sampling calculations.

Chapter 3

Sampling errors in nested sampling parameter estimation

This chapter provides the first explanation of the two main sources of sampling errors in nested sampling parameter estimation, and presents a new diagrammatic representation for the process. We find no previously existing method can accurately measure the parameter estimation errors of a single nested sampling run, and propose a method for doing so using a new algorithm for dividing nested sampling runs. The material presented is an edited version of Higson et al. (2018).

3.1 Introduction

Sampling errors in nested sampling parameter estimation differ from those in Bayesian evidence calculation, but have been little studied in the literature. As a result these errors are poorly understood compared to other numerical methods, despite the growing popularity of the technique and its widespread use in astronomy.

Correctly quantifying uncertainty due to sampling errors is vital for identifying spurious results. Conversely, finding such errors are very small may imply an unnecessarily large amount of computational resource is being used for the calculation. This chapter has two goals: to provide an explanation of the sources of these errors and an empirical technique for estimating them. One obvious method is to repeat the analysis a number of times, although this increases the computational cost by a corresponding factor. Interestingly, we find no current method can accurately estimate these errors

on parameter estimates from a single analysis, and so we present a new method for doing this. Our approach uses a new algorithm for dividing a single nested sampling run into multiple valid nested sampling runs; these can then be recombined in different combinations using resampling techniques such as the bootstrap. We test our results and new method empirically.

The chapter begins with background on sampling errors in parameter estimation from posterior samples. We then explain the two main sources of sampling errors in nested sampling parameter estimation in Section 3.3, and present a new diagrammatic representation of the process (illustrated in Figures 3.2a to 3.2e). Section 3.4 describes our new method for measuring sampling errors from a single nested sampling run, using our new algorithm for division of such runs.

We empirically test our method's accuracy in Section 3.5 with the help of analytical cases in the manner described by Keeton (2011). Here one can obtain uncorrelated samples from the prior space within some likelihood contour using standard techniques, and we term the resulting procedure *perfect nested sampling*. Results in Section 3.5 were calculated using an early version of the `perfectns` software package (Higson, 2018c). In Section 3.6 we test sampling error estimates from our method for `PolyChord` calculations, and Section 3.7 describes the application of these sampling error estimates to gravitational waves in Chua et al. (2018). Our approach accurately quantifies uncertainties on parameter estimates from the stochasticity of the nested sampling algorithm, but software used for practical problems may produce additional errors from correlated samples within likelihood contours that are specific to a given implementation — these are discussed in detail in Chapter 4. Our method gives superior performance to the current approach and can be easily be applied to existing nested sampling software; we have implemented it in the `nestcheck` software package (Higson, 2018b).

3.2 Background: sampling errors in parameter estimation

Sampling techniques such as nested sampling provide information about a posterior distribution $\mathcal{P}(\boldsymbol{\theta})$ by producing a set of weighted samples

$$\mathcal{S} = \{(\boldsymbol{\theta}_s, p_s), s = 1, \ldots, n_{\mathrm{samp}}\}, \tag{3.1}$$

where each $\boldsymbol{\theta}_s$ is drawn from $\mathcal{P}(\boldsymbol{\theta})$ with probability proportional to $p_s \times \mathcal{P}(\boldsymbol{\theta}_s)$, and $\sum_{s\in\mathcal{S}} p_s = 1$. Numerical results are then computed from the samples $\mathcal{S}$. For example, the posterior expectation of a function of the parameters $f(\boldsymbol{\theta})$ can be estimated as

$$\mathrm{E}[f(\boldsymbol{\theta})] = \int f(\boldsymbol{\theta})\mathcal{P}(\boldsymbol{\theta})\, d\boldsymbol{\theta} \approx \sum_{s\in\mathcal{S}} p_s f(\boldsymbol{\theta}_s). \tag{3.2}$$

In this case the sampling error is the difference between $\sum_{s\in\mathcal{S}} p_s f(\boldsymbol{\theta}_s)$ and the exact value of $\mathrm{E}[f(\boldsymbol{\theta})]$. Often the posterior distributions of parameters $\boldsymbol{\theta}$ are of interest, and are estimated numerically from the samples by dividing the parameter space into cells or via kernel density estimation.

There have been many works on approximating MCMC sampling errors, including investigation of quantiles and the amount of computation required to reach some level of accuracy — see for example Doss et al. (2015), Flegal et al. (2008) and Liu et al. (2016). In particular Sequential Monte Carlo samplers (mentioned in Section 2.3) have similarities with nested sampling, and their sampling errors are better understood. For some related methods such as the Tootsie Pop algorithm (Huber and Schott, 2014) and accelerated simulated annealing (Bezáková et al., 2008) the error distribution is known exactly, although these techniques are less widely used. This chapter introduces empirically tested techniques for quantifying sampling errors from the nested sampling algorithm.

In nested sampling parameter estimation, the sample weights (2.20) present a departure from traditional sampling approaches in that the $w_i(\mathbf{t})$ are random variables, with their stochasticity determined by (2.18). When computing expectations (3.2) there is now an additional error associated with our lack of knowledge of the precise values $p_i(\mathbf{t})$. Nested sampling software packages such as `MultiNest` and `PolyChord` produce posterior files containing only the expected values

$$\mathrm{E}[p_i(\mathbf{t})] = \frac{\mathrm{e}^{-i/n}\mathcal{L}_i}{\sum_j \mathrm{e}^{-j/n}\mathcal{L}_j}. \tag{3.3}$$

To account for the stochasticity in the weights p_i, Skilling (2006) suggests simulating the prior volume shrinkage ratios $\mathbf{t}$ in the same manner as for evidence estimation (mentioned in Section 2.4.1), and using these simulations to calculate a set of values for estimators such as (3.2). The sampling error should then be estimated from the variation within this sample; we term this the *simulated weights method*. We believe

this procedure is the only estimate of sampling errors in parameter estimation from a single nested sampling run proposed in the literature. However it is in general an *underestimate*, as can be seen in the numerical tests in Section 3.5. Appendix 3.B discusses this underestimation of errors in detail.

We now describe why the simulated weights method does not capture all sources of sampling errors, and in Section 3.4 we propose a new method for correctly computing these errors.

3.3 Sources of sampling errors in nested sampling parameter estimation

In order to understand why the simulated weights method underestimates sampling errors, we require a result from Chopin and Robert (2010). They show that the expectation integral (3.2) may be re-phrased in terms of the prior volume X via:

$$\mathrm{E}[f(\boldsymbol{\theta})] = \int f(\boldsymbol{\theta})\mathcal{P}(\boldsymbol{\theta})\, d\boldsymbol{\theta} = \int f(\boldsymbol{\theta})\frac{\mathcal{L}(\boldsymbol{\theta})\pi(\boldsymbol{\theta})}{\mathcal{Z}}\,\mathrm{d}\boldsymbol{\theta} = \frac{1}{\mathcal{Z}}\int \tilde{f}(X)\mathcal{L}(X)\,\mathrm{d}X, \tag{3.4}$$

where $\tilde{f}(X)$ is the prior expectation of $f(\boldsymbol{\theta})$ on some iso-likelihood contour $\mathcal{L}(\boldsymbol{\theta}) = \mathcal{L}(X)$,

$$\tilde{f}(X) \equiv \mathrm{E}^{\pi}[f(\boldsymbol{\theta})|\mathcal{L}(\boldsymbol{\theta}) = \mathcal{L}(X)]. \tag{3.5}$$

The simulated weights approach amounts to discretising the integral (3.4) as

$$\frac{1}{\mathcal{Z}}\int \tilde{f}(X)\mathcal{L}(X)\,\mathrm{d}X \approx \frac{1}{\mathcal{Z}}\sum_i \tilde{f}(X_i)\,\mathcal{L}_i\,\frac{1}{2}(X_{i-1} - X_{i+1}), \tag{3.6}$$

and, most importantly, further requiring that we may use $f(\boldsymbol{\theta}_i)$ as a proxy for $\tilde{f}(X_i)$ at each point X_i. In some special cases $f(\boldsymbol{\theta}_i) = \tilde{f}(X_i)$ for all $\boldsymbol{\theta}$ and this approach is valid, for example when $f(\boldsymbol{\theta}_i) = \tilde{f}(X_i) \propto -\log \mathcal{L}_i$ (entropy computation), but in general it is not. This can cause significant inaccuracies as iso-likelihood contours often span wide ranges of different parameter values, as illustrated in Figure 3.1 (based on Figure 1 in Handley et al., 2015a). There are also some errors from discretising the integral in (3.6) using the trapezium rule, but these are typically small unless the number of live points n is low. Furthermore, errors due to the truncation of the sum in (3.6) when the algorithm terminates can be made negligible with appropriate termination conditions.

To summarise, the dominant sampling errors in estimating some parameter or function of parameters from perfect nested sampling typically come from two sources:

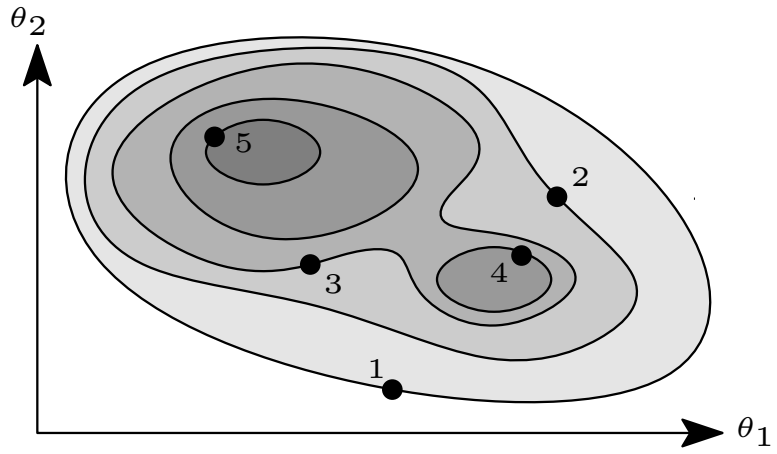


Figure 3.1: Nested sampling dead points and iso-likelihood contours for a two-dimensional multi-modal likelihood $\mathcal{L}(\boldsymbol{\theta})$; darker shading shows higher likelihoods. Iso-likelihood contours can pass through a wide range of different parameter values.

(i) approximating the unknown prior volumes $w_i(\mathbf{t})$ with their expectation $\mathrm{E}[w_i(\mathbf{t})]$ using (2.18);

(ii) approximating the mean value of a function of parameters over an entire iso-likelihood contour $\tilde{f}(X_i)$ with its value at a single point $f(\boldsymbol{\theta}_i)$.

Errors from (i) are also present in evidence calculation; in the parameter estimation case they are typically smaller as results depend only on the *relative* weights of the samples. In contrast (ii) is only present in parameter estimation, where it is typically a significant or dominant source of sampling errors. The relative contributions of (i) and (ii) are empirically tested in Appendix 3.A, where they are calculated for analytical cases by using exact values for weights $w_i(\mathbf{t})$ and by replacing $f(\boldsymbol{\theta}_i)$ with $\tilde{f}(X_i)$. The simulated weights method underestimates sampling errors in parameter estimation as it ignores errors from (ii).

We now introduce a new diagrammatic representation of nested sampling parameter estimation to illustrate the two different sources of sampling errors.

3.3.1 Diagrammatic representation

Nested sampling transforms evidence calculations of any dimension into a 1-dimensional problem[1] in $\mathcal{L}(X)$ which can be entirely represented on a diagram like Figure 2.1. An analogous diagram for parameter estimation must also illustrate sampling a single point $f(\boldsymbol{\theta}_i)$ on each iso-likelihood contour $\mathcal{L}(\boldsymbol{\theta}) = \mathcal{L}(X_i)$ from the distribution $P(f(\boldsymbol{\theta})|X_i)$.

We propose a generalisation of Figure 2.1 for visualising parameter estimation problems, and present it in Figures 3.2a to 3.2e. The top panel in each figure is similar to Figure 2.1 and shows the relative posterior mass $\mathcal{L}(X)X$ at each value of $\log X$. The lower central panel shows the probability distribution $P(f(\boldsymbol{\theta})|X)$ and its mean $\tilde{f}(X)$. The posterior distribution is shown on the left — this is equal to the distributions $P(f(\boldsymbol{\theta})|X)$ (the lower central panel) marginalised over X in proportion to the posterior weight at each X (the top panel).

For these example plots we use d-dimensional spherical unit Gaussian likelihoods

$$\mathcal{L}(\boldsymbol{\theta}) = (2\pi)^{-d/2}\mathrm{e}^{-|\boldsymbol{\theta}|^2/2} \tag{3.7}$$

and d-dimensional spherical unit Cauchy likelihoods

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{\Gamma(\frac{1+d}{2})}{\pi^{(d+1)/2}}\left(1 + |\boldsymbol{\theta}|^2\right)^{-(\frac{d+1}{2})}, \tag{3.8}$$

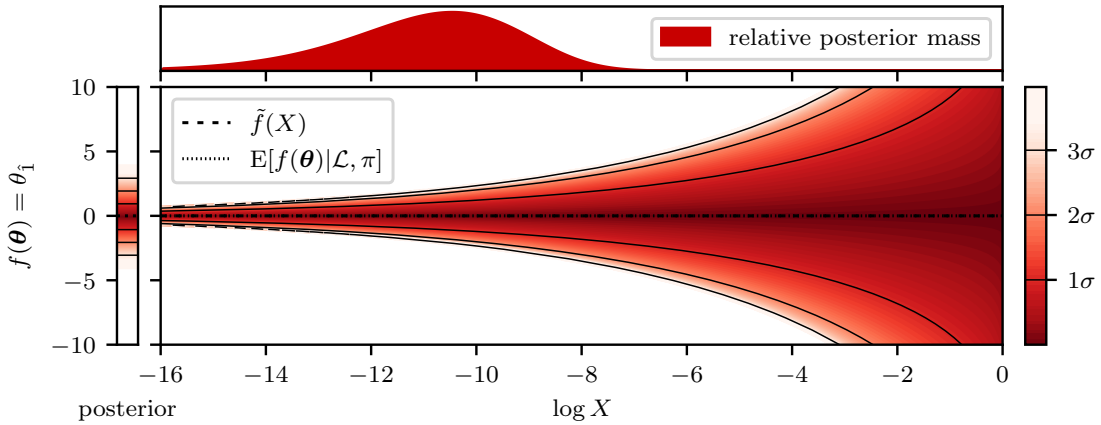
with d-dimensional co-centred spherical Gaussian priors

$$\pi(\boldsymbol{\theta}) = (2\pi\sigma_\pi^2)^{-d/2}\mathrm{e}^{-|\boldsymbol{\theta}|^2/2\sigma_\pi^2}. \tag{3.9}$$

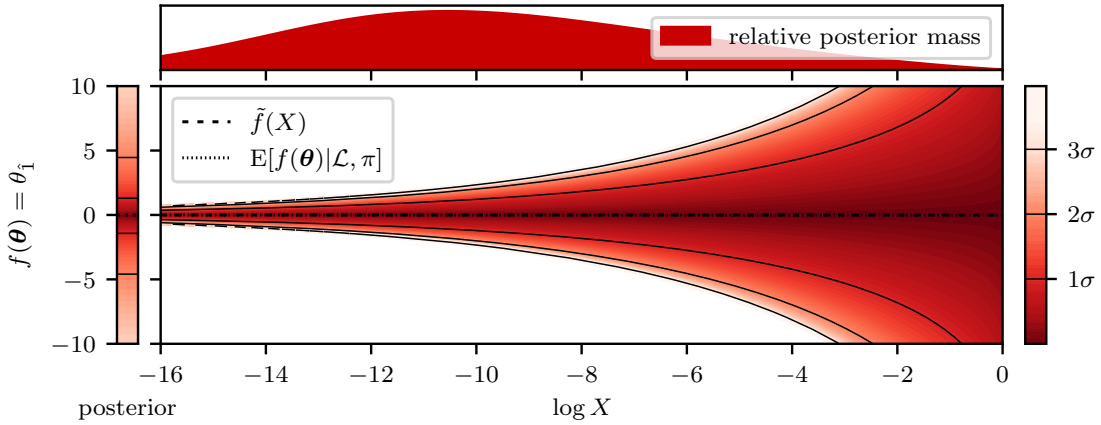
Here Γ denotes the gamma function, and in this chapter all Gaussian priors (3.9) use $\sigma_\pi = 10$. We denote the first component of the $\boldsymbol{\theta}$ vector as $\theta_{\hat{1}}$, although by symmetry the results will be the same for any component. $\overline{\theta_{\hat{1}}}$ and $\overline{\theta_{\hat{1}}^2}$ are the first and second moments of the posterior distribution of $\theta_{\hat{1}}$.

The form of the distribution $P(f(\boldsymbol{\theta})|X)$ as X varies depends on the likelihood only through the shape of the iso-likelihood contours $\mathcal{L}(\boldsymbol{\theta}) = \mathcal{L}(X)$. Therefore the lower central panel of the diagrams for some $f(\boldsymbol{\theta})$ is the same for any likelihoods with the same contours — this can be seen in Figures 3.2a and 3.2b, where the differences in

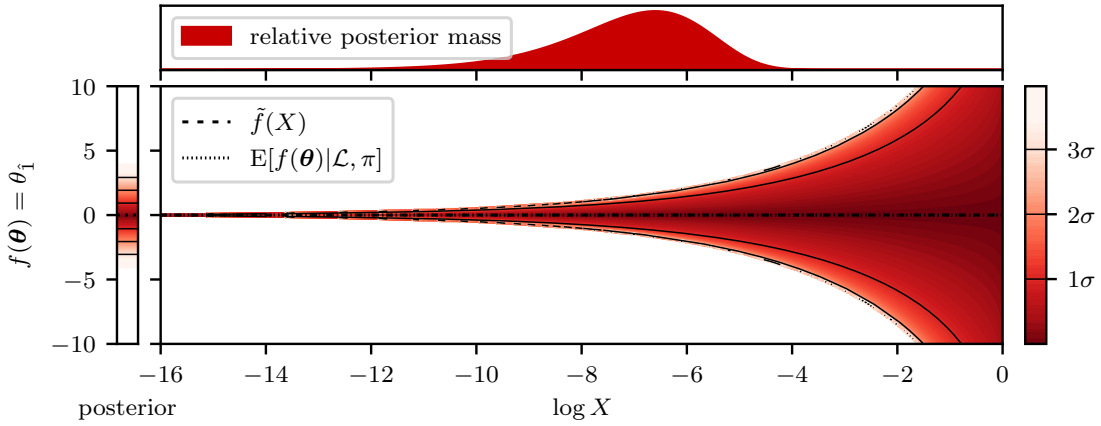
[1] For practical nested sampling problems implementation-specific errors can differ for two likelihoods with the same $\mathcal{L}(X)$. For example if one likelihood has a much higher dimension and a much larger number of modes than the other, it may have larger errors from the software failing to explore the parameter space fully.



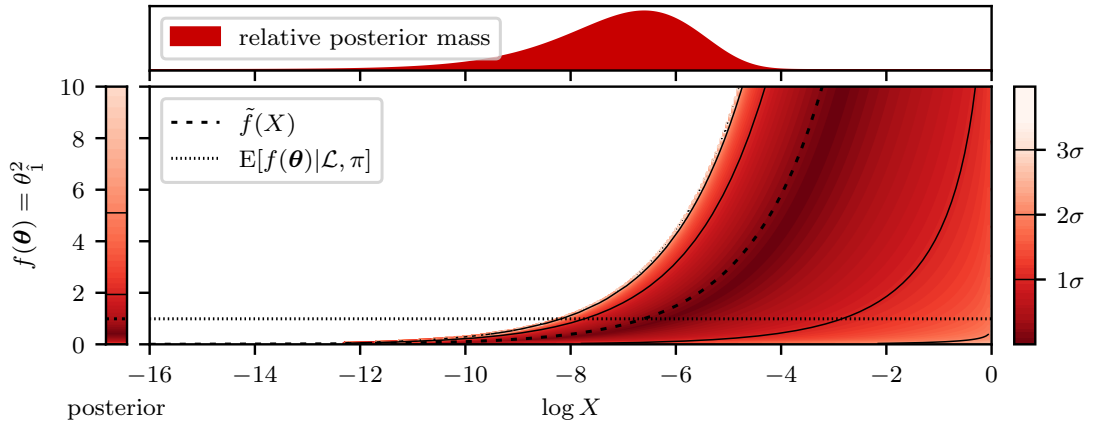
(a) $f(\boldsymbol{\theta}) = \theta_{\hat{1}}$ with a 5-dimensional Gaussian likelihood (3.7) and a Gaussian prior (3.9).



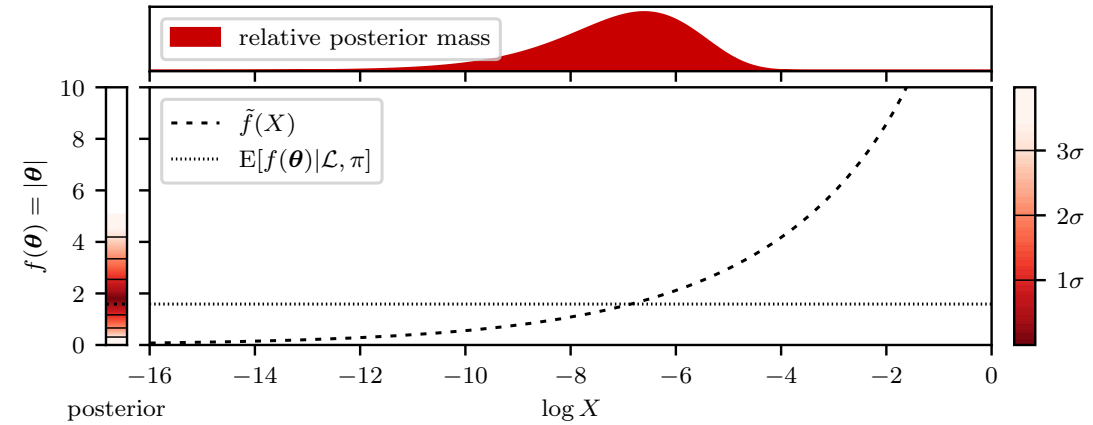
(b) $f(\boldsymbol{\theta}) = \theta_{\hat{1}}$ with a 5-dimensional Cauchy likelihood (3.8) and a Gaussian prior (3.9).



(c) $f(\boldsymbol{\theta}) = \theta_{\hat{1}}$ with a 3-dimensional Gaussian likelihood (3.7) and a Gaussian prior (3.9).



(d) $f(\boldsymbol{\theta}) = {\theta_1}^2$ with a 5-dimensional Gaussian likelihood (3.7) and a Gaussian prior (3.9).



(e) $f(\boldsymbol{\theta}) = |\boldsymbol{\theta}|$ (i.e. the radial distance from the likelihood's maximum) with a 5-dimensional Gaussian likelihood (3.7) and a Gaussian prior (3.9). In this case $f(\boldsymbol{\theta}_i) = \tilde{f}(X_i)$ for all $\boldsymbol{\theta}$ and sampling errors are only from uncertainty in prior volume shrinkages and the trapezium rule approximation.

Figure 3.2: Nested sampling parameter estimation diagrams: in each case the top panel shows the relative posterior mass at each value of $\log X$ ($\propto \mathcal{L}(X)X$). The lower central panel shows the distribution $P(f(\boldsymbol{\theta})|X)$ of values $f(\boldsymbol{\theta})$ on each iso-likelihood contour $\mathcal{L}(\boldsymbol{\theta}) = \mathcal{L}(X)$; the dashed line shows the expectation of this distribution which we defined in (3.5) as $\tilde{f}(X)$. The left panel shows the posterior distribution of $f(\boldsymbol{\theta})$, with the dotted line showing its posterior expectation. The colour scale shows the fraction of the cumulative probability distribution lying between some region and the median.

the posterior (left panel) are due only to the different posterior weights in $\log X$ (top panel). Adapted versions of these diagrams which can be easily made from the samples produced by nested sampling software for *a priori* unknown likelihoods are introduced in Chapter 4.

3.3.2 Transforming a parameter estimation problem into 2 dimensions

As illustrated by our diagrams, nested sampling parameter estimation is fundamentally a 2-dimensional problem in $\mathcal{L}(X)$ and $P(f(\boldsymbol{\theta})|X)$. In fact a perfect nested sampling parameter estimation calculation for some $f(\boldsymbol{\theta})$ given $\mathcal{L}(\boldsymbol{\theta})$ is equivalent to a 2-dimensional problem for $f^*(\boldsymbol{\theta}^*)$ given $\mathcal{L}^*(\boldsymbol{\theta}^*)$ when

$$\mathcal{L}^*(\boldsymbol{\theta}^*) = \mathcal{L}(X), \tag{3.10}$$

$$P(f^*(\boldsymbol{\theta}^*)|X) = P(f(\boldsymbol{\theta})|X), \tag{3.11}$$

for all X. Any transformation satisfying (3.10) and (3.11) will leave our proposed diagram for the calculation unchanged. Parameter estimation can also be represented as a 1-dimensional problem in $\mathcal{L}^*(\boldsymbol{\theta}^*) = \mathcal{L}(X)$ combined with a univariate stochastic process for each dead point i with the distribution $P(f(\boldsymbol{\theta})|X_i)$.

One way to express a general nested sampling calculation in 2 dimensions is to map it onto the unit square $\boldsymbol{\theta}^* = (X, Y)$ with uniform priors $X, Y \in [0, 1]$ and a likelihood $\mathcal{L}^*(\boldsymbol{\theta}^*) = \mathcal{L}(X)$ which is independent of Y and satisfies (3.10). In this case X is, as before, the remaining fractional prior volume and Y parameterises each iso-likelihood contour. Using inverse transform sampling, for a general $f(\boldsymbol{\theta})$ a corresponding $f^*(\boldsymbol{\theta}^*)$ satisfying (3.11) is

$$f^*(\boldsymbol{\theta}^*) = f^*(X, Y) = F^{-1}(Y|X), \tag{3.12}$$

where $F^{-1}(Y|X)$ is the inverse of the cumulative distribution

$$F(Y|X) = \int_{-\infty}^{Y} P(f(\boldsymbol{\theta}) = h|X)\,\mathrm{d}h. \tag{3.13}$$

As an example let us consider d-dimensional spherically symmetric likelihoods such as (3.7) or (3.8) with co-centred spherically symmetric priors such as (3.9). Then $X(\boldsymbol{\theta})$ is a function only of the radial distance from the centre $|\boldsymbol{\theta}|$, and the iso-likelihood contours $\mathcal{L}(\boldsymbol{\theta}) = \mathcal{L}(X)$ are hyperspherical shells of some radius $|\boldsymbol{\theta}|_X$. The probability distribution

of a single parameter $\theta_{\hat{1}}$ (a single component of $\boldsymbol{\theta}$) on such an iso-likelihood contour is then

$$P(\theta_{\hat{1}}|X) = \begin{cases} \frac{\Gamma\left(\frac{d}{2}\right)}{|\boldsymbol{\theta}|_X \, \Gamma(\frac{1}{2}) \, \Gamma\left(\frac{d-1}{2}\right)} \left(1 - \frac{\theta_{\hat{1}}^2}{|\boldsymbol{\theta}|_X^2}\right)^{\frac{d-3}{2}} & \text{if } -|\boldsymbol{\theta}|_X < \theta_{\hat{1}} < |\boldsymbol{\theta}|_X, \\ 0 & \text{otherwise.} \end{cases} \tag{3.14}$$

$\theta_{\hat{1}}$ can be sampled directly or used to calculate the inverse cumulative distribution which together with knowledge of the function $\mathcal{L}(X)$ allows the parameter estimation of a d-dimensional Gaussian to be transformed into a 2-dimensional problem on the unit square.

Samples from (3.14) can be generated efficiently using the symmetry around $\theta_{\hat{1}} = 0$ and the change of variables $\Theta = \theta_{\hat{1}}^2 / |\boldsymbol{\theta}|_X^2$ to give a Beta distribution

$$P(\Theta|X) = \begin{cases} \frac{\Gamma\left(\frac{d}{2}\right)}{\Gamma\left(\frac{1}{2}\right) \Gamma\left(\frac{d-1}{2}\right)} \Theta^{-\frac{1}{2}} (1-\Theta)^{\frac{d-3}{2}} & \text{if } 0 < \Theta < 1, \\ 0 & \text{otherwise,} \end{cases} \tag{3.15}$$

$$\Theta \sim \text{Beta}\left(\frac{1}{2}, \frac{d-1}{2}\right). \tag{3.16}$$

This technique is used for the numerical tests in Section 3.5 with `perfectns`, and allows the efficient sampling of high dimensional spherically symmetric distributions where only a few parameters are of interest without generating all the remaining uninteresting parameters.

3.4 Estimating sampling errors in nested sampling parameter estimation

Following the discussion of sources of sampling errors in Section 3.3, we seek a method for correctly calculating parameter estimation sampling errors from a single nested sampling run. As no additional samples $\boldsymbol{\theta}_i$ are available, a natural starting point is to utilise resampling techniques such as the jackknife (Tukey, 1958), bootstrap (Efron, 1979) and Bayesian bootstrap (Rubin, 1981), which estimate the uncertainty on inferences from a set of samples by calculating the variation when samples are re-weighted.

However, as described in Section 2.4.1, the uncertainty in nested sampling weights $w_i(\mathbf{t})$ produces additional sampling errors which are unique to the nested sampling

process. These are not accounted for by naïvely applying jackknives and bootstraps to posterior samples produced by nested sampling, and these approaches fail when tested numerically. We instead require a method for dividing runs in a manner that preserves the statistical properties of nested sampling. No such method exists in the literature, so we present one in the remainder of this section.

3.4.1 Dividing runs into threads

Skilling (2006) describes how several nested sampling runs $r = 1, 2, \ldots$ with $n^{(r)}$ live points may be combined simply by merging the dead points and sorting by likelihood value. The combined sequence of dead points is equivalent to a single nested sampling run with $n = \sum_r n^{(r)}$ live points.

In fact, as we show now, the reverse procedure is also possible. A nested sampling run with n points can be unwoven into a set of n valid nested sampling runs, each with $n^{(r)} = 1$. We term these single live point runs *threads*. During nested sampling, each dead point i is replaced by a new point sampled uniformly within its iso-likelihood contour $\mathcal{L}(\boldsymbol{\theta}) = \mathcal{L}_i$. Starting from each initial live point that is generated, one may follow this sequence of replacements down the set of dead points. This sub-sequence of dead points is in fact a nested sampling run with $n = 1$. Our algorithm for dividing a nested sampling run into its constituent threads is presented more formally in Algorithm 2.

Result: n threads.
Data: Dead points and the iterations at which they were sampled for a nested sampling run with n live points.
Rank dead points by likelihood in ascending order;
while $i \in n$ **do**
    make a new stack i;
    select one of the initial points sampled at the start of the run;
    move the point out to the stack i;
    while *iteration* $<$ *final iteration* **do**
        select point sampled at the iteration where previous point was replaced ("died");
        move the point to the stack i;
    end
end

Algorithm 2: Splitting a nested sampling run into threads.

A few points are worthy of note:

1. splitting a run by randomly selecting some fraction of the dead points will not produce threads (i.e. single point nested sampling runs);

2. one may split a given nested sampling run into separate runs with $n^{(r)} \neq 1$ by first separating into threads, and then recombining threads as desired;

3. the algorithm can be easily adapted for varying numbers of live points by permitting it to select multiple points on contours where n increases. This can result in constituent threads stopping or dividing into multiple threads part of the way through the run;

4. typically there is only one point which was sampled uniformly from the prior volume within each dead point i's iso-likelihood contour $\mathcal{L}(\boldsymbol{\theta}) = \mathcal{L}_i$ — the point which replaced i. A sufficient condition for a nested sampling run to only have one unique division into threads is that $\mathcal{L}(X)$ is an injective function;

5. in order for the threads to be true nested sampling runs, care must be taken with the termination conditions conditions used. See Appendix 3.D for a full discussion.

Given that threads represent independent nested sampling runs, one may apply standard resampling techniques to the set of threads and approximate the entire sampling error distribution without making assumptions about its form. This works as the $\log X_i$ values of the dead points i from some run with n live points form a Poisson process with rate n, meaning the $\log X_j$ values of the dead points j of a single thread are a Poisson process of rate 1. For typical problems with computationally expensive likelihoods the computational cost of even a large number of resampling replications is negligible.

Having introduced a framework for applying resampling to nested sampling parameter estimation we now present an example method using bootstrap resampling.

3.4.2 Bootstrap estimate of sampling errors

Given n observations $\mathbf{x} = (x_1, \ldots, x_n)$, the bootstrap (Efron, 1979) creates new data sets $\mathbf{x}_b^*$ by drawing n samples from $\mathbf{x}$ with replacement. This corresponds to approximating the probability distribution of a single data point x as

$$P(x) \approx \frac{1}{n} \sum_{i=1}^{n} \delta(x - x_i), \tag{3.17}$$

where $\delta(x)$ is the Dirac delta function (Ivezić et al., 2014).

As the form of the distribution of sampling errors for a general nested sampling parameter estimation problem is not known, we use the non-parametric bootstrap. In this case the uncertainty on a quantity $T(\mathbf{x})$ calculated from the data can be estimated by calculating $T(\mathbf{x}_b^*)$ for a number of resampled data sets $b = 1, \ldots, B$. For example the bootstrap estimate of the standard error on $T(\mathbf{x})$ is

$$\text{St.Dev.}[T(\mathbf{x})] = \sqrt{\frac{1}{B-1}\sum_{b=1}^{B}\left(T(\mathbf{x}_b^*) - \overline{T(\mathbf{x}_b^*)}\right)^2}, \quad \text{where } \overline{T(\mathbf{x}_b^*)} = \frac{1}{B}\sum_{b=1}^{B} T(\mathbf{x}_b^*). \tag{3.18}$$

There are many methods for calculating approximate credible intervals on $T(\mathbf{x})$ from bootstrap replications $\{T(\mathbf{x}_b^*)\}$ — see Efron and Tibshirani (1986) for a detailed discussion. A simple approach from Johnson (2001) is to estimate the boundaries of the $100\alpha\%$ and $100(1-\alpha)\%$ credible regions[2] as

$$\text{C.I.}_{100\alpha\%}\left(T(\mathbf{x})\right) = 2T(\mathbf{x}) - G^{-1}(1-\alpha), \tag{3.19}$$

$$\text{C.I.}_{100(1-\alpha)\%}\left(T(\mathbf{x})\right) = 2T(\mathbf{x}) - G^{-1}(\alpha), \tag{3.20}$$

where $G^{-1}(x)$ is the inverse cumulative distribution of the bootstrap samples $\{T(\mathbf{x}_b^*)\}$. $B = 50$ is typically sufficient for an estimate of the standard deviation of a parameter estimate due to sampling errors, but depending on the method used credible intervals on parameter estimates may require 1,000 bootstrap replications or more (Efron and Tibshirani, 1986).

When the bootstrap is applied to nested sampling each observation x_i is a thread, and the number of observations is n. Calculating the quantity $T(\mathbf{x})$ involves first combining the set of threads $\mathbf{x}$ into a single run using Skilling (2006)'s method (described in Section 3.4.1), then performing a standard nested sampling calculation including estimating the weight of each point $w_i(\mathbf{t})$ statistically. Including the same thread multiple times does not cause problems — repeated dead points $\boldsymbol{\theta}_i = \boldsymbol{\theta}_{i+1}$ are simply assigned the weights $w_i(\mathbf{t})$ and $w_{i+1}(\mathbf{t})$ respectively. Algorithm 3 provides a set of bootstrap replications and an estimate of the standard deviation of sampling errors.

We find that bootstrap resampling gives better results than jackknife resampling, which fails to calculate sampling errors on credible intervals of posterior distributions

[2] If the distribution of bootstrap replications $T(\mathbf{x}_b^*)$ is skewed then the implied probability distribution of T is skewed in the opposite direction, as can be seen from (3.19) and (3.20). See Loredo (2012, Section 2) for a discussion.

Result: Sampling errors and bootstrap replications for the nested sampling calculation $T(\text{dead points}, \text{weights})$.
Data: List of dead points and the steps they were sampled at.
Divide dead points into a list of threads $\mathbf{x}$ using Algorithm 2;
while $b \in B$ **do**
  create a list of n threads $\mathbf{x}_b^*$ by sampling $\mathbf{x}$ with replacement;
  calculate $T(\mathbf{x}_b^*) \equiv T(\text{dead points}_b^*, \text{weights}_b^*)$;
end
calculate St.Dev.$[T(\mathbf{x})] = \sqrt{\frac{1}{B-1}\sum_{b=1}^{B}\left(T(\mathbf{x}_b^*) - \overline{T(\mathbf{x}_b^*)}\right)^2}$.

Algorithm 3: Bootstrap sampling error calculation.

of parameters such as C.I.$_{84\%}(\overline{\theta_{\hat{1}}})$. The Bayesian bootstrap was not used as it gives each observation a non-integer weight, which requires modifying nested sampling's use of dead points to statistically estimate prior volume shrinkages.

Resampling techniques such as the bootstrap can generate many simulated runs with the same number of live points n as the original run. In comparison sampling error estimates from simply splitting a run into many smaller runs and assessing their variation perform poorly, as shown in Appendix 3.C.

3.5 Numerical tests

Following Keeton (2011) we first test our new method using analytic cases where uncorrelated samples can be easily obtained from the prior within an iso-likelihood contour, allowing us to perform perfect nested sampling. This ensures our results are not affected by imperfect implementation of the nested sampling algorithm by a specific software. These numerical tests used an earlier version of the `perfectns` software package.

As discussed in Section 3.3, perfect nested sampling parameter estimation problems depend on the likelihood $\mathcal{L}(\boldsymbol{\theta})$ and prior $\pi(\boldsymbol{\theta})$ only through the distribution of posterior mass $\mathcal{L}(X)$ and the distribution of parameters on iso-likelihood contours $P(f(\boldsymbol{\theta})|X)$, both of which are functions of both $\mathcal{L}(\boldsymbol{\theta})$ and $\pi(\boldsymbol{\theta})$. We therefore empirically test our method using a wide range of distributions of posterior mass, and examine several functions of parameters $f(\boldsymbol{\theta})$ in each case. We construct such tests using Gaussian likelihoods (3.7) and Cauchy likelihoods (3.8) of a variety of dimensions d, each with a Gaussian prior (3.9). The different distributions of posterior mass for different d

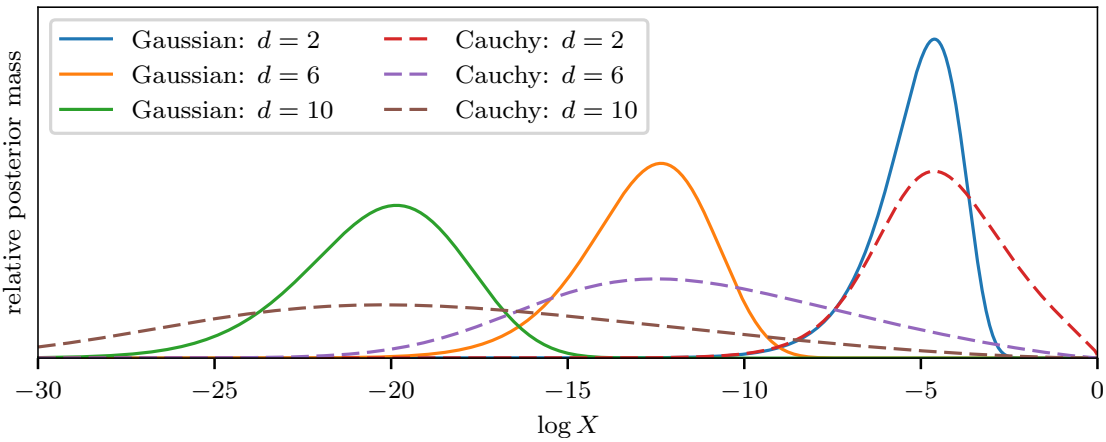


Figure 3.3: Relative posterior mass as a function of $\log X$ ($\propto \mathcal{L}(X)X$) for Gaussian likelihoods (3.7) and Cauchy likelihoods (3.8) of different dimensions d with Gaussian priors (3.9). The lines are scaled so that the area under each of them is equal.

are illustrated in Figure 3.3; the Cauchy distributions have extremely fat tails with significant sample weights throughout the range of $\log X$ values explored.

We use the termination conditions described by Handley et al. (2015b, Section 3.4), stopping when the estimated evidence contained in the live points is less than 10^{-4} times the evidence contained in dead points (see Appendix 3.D for a discussion of termination conditions for nested sampling parameter estimation). Numerical calculations for high-dimensional cases are performed in two dimensions using the technique described in Section 3.3.2.

As in Section 3.3 we denote the first component of the $\boldsymbol{\theta}$ vector as $\theta_{\hat{1}}$, although by symmetry the results will be the same for any component. $\overline{\theta_{\hat{1}}}$ and $\overline{\theta_{\hat{1}}^2}$ are the first and second moments of the posterior distribution of $\theta_{\hat{1}}$, and the one-tailed $Y\%$ upper credible interval $\text{C.I.}_{Y\%}(\overline{\theta_{\hat{1}}})$ is the value $\theta^*_{\hat{1}}$ for which $P(\theta_{\hat{1}} < \theta^*_{\hat{1}}|\mathcal{L}, \pi) = Y/100$.

3.5.1 3-dimensional Gaussian example

We first test our bootstrap approach to estimating sampling errors on a 3-dimensional Gaussian likelihood (3.7) — Figure 3.4 illustrates sampling errors on the posterior distributions of parameters in this case. Unlike the simulated weights method, the mean estimates of sampling errors from our method are very close to measurements of sampling errors from repeated calculations — this is shown in the second row of Table 3.1. Furthermore the fractional variation of estimates from single runs around the mean es-

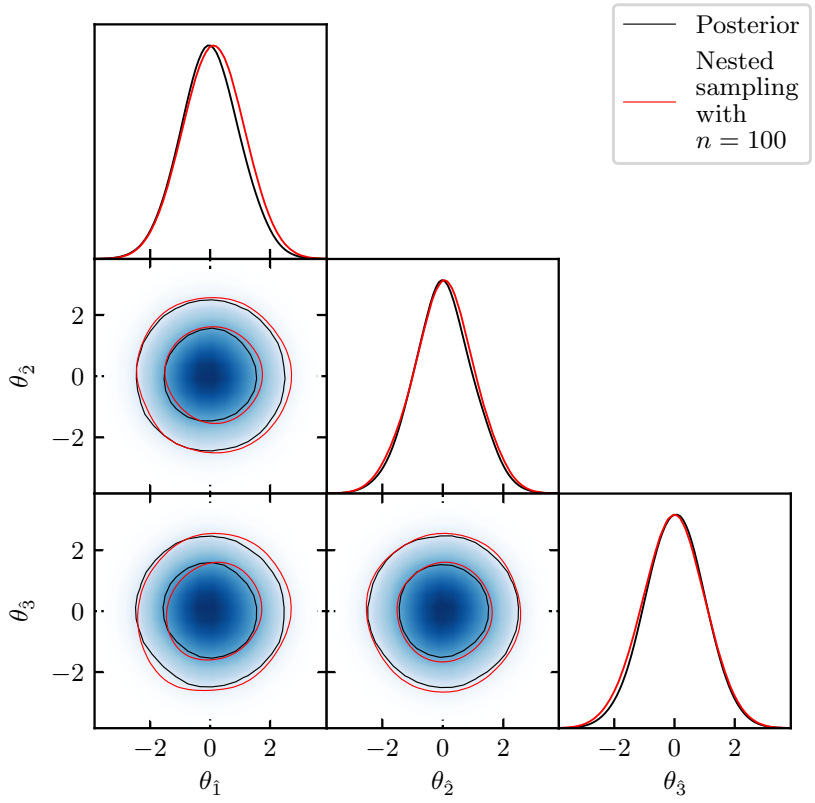


Figure 3.4: Sampling errors in a perfect nested sampling calculation for a 3-dimensional Gaussian likelihood (3.7) and a uniform prior. The shading and black lines show the analytic posterior distribution and the 68% and 95% credible intervals. The red lines show the calculated posterior credible intervals for a nested sampling run with $n = 100$, and differ from the analytic answer due to sampling errors.

timate is similar to that from the simulated weights method, as shown in the fourth and fifth rows, indicating our method will give a reasonable estimate of sampling errors when only a single nested sampling run is available.

The final two rows of Table 3.1 show the empirical coverage rates for bootstrap credible intervals are very close to their nominal values. Figure 3.5 shows estimates of the full sampling error distribution from a single run nested sampling run using the bootstrap and simulated weights methods; the bootstrap results are much closer to the sampling errors observed in repeated calculations, and give accurate estimates of the 1σ and 2σ credible intervals.

Appendix 3.E shows similar numerical tests for a 3-dimensional Cauchy likeli-

	$\overline{\theta_{\hat{1}}}$	$\overline{\theta_{\hat{1}}^2}$	$\text{C.I.}_{84\%}(\overline{\theta_{\hat{1}}})$
Repeated runs St.Dev.	0.032(0.2)	0.050(0.4)	0.055(0.4)
BS St.Dev. / Repeats St.Dev.	1.003(7)	0.998(7)	1.008(8)
Sim St.Dev. / Repeats St.Dev.	0.715(5)	0.882(6)	0.785(7)
BS St.Dev. estimate variation	7.5(1)%	8.6(1)%	17.7(3)%
Sim estimate variation	6.0(1)%	7.3(1)%	19.7(3)%
$\text{BS C.I.}_{95\%}$	0.053(3)	1.080(5)	1.077(7)
BS Mean±1St.Dev. coverage	68.4%	68.2%	68.9%
$\text{BS C.I.}_{95\%}$ coverage	95.0%	93.4%	93.1%

Table 3.1: Sampling errors for a 3-dimensional Gaussian likelihood (3.7), a Gaussian prior (3.9) and $n = 200$. The first row shows the standard deviation of 10,000 nested sampling calculations. The second and third rows show the mean of 2,000 error estimates from the bootstrap and simulated weights methods respectively as a ratio to the error observed from repeated calculations; 200 weight simulations and 200 bootstrap replications were used for each run. The fourth and fifth rows show the standard deviations of sampling error estimates for both methods as a percentage of the mean estimate. The sixth row shows the mean of 100 bootstrap estimates of the one-tailed 95% credible interval on the calculation result given the sampling error, each using 1,000 bootstrap replications. The final two rows show the empirical coverage of the bootstrap standard error and 95% credible interval from the 10,000 repeated calculations. Numbers in brackets show the error on the final digit.

hood (3.8). Even for this challenging, fat-tailed distribution our method performs similarly to the Gaussian case, giving accurate mean error estimates and estimates of credible intervals with measured coverage similar to their nominal coverage.

3.5.2 Sampling errors in different dimensions

We now verify the bootstrap method's accuracy for Gaussian (3.7) and Cauchy (3.8) likelihoods of between 2 and 50 dimensions. Figure 3.6 shows bootstrap sampling error estimates accurately match the errors measured from repeated calculations, even for the challenging fat-tailed Cauchy distribution. In contrast the simulated weights method consistently underestimates the sampling errors in parameter estimation, although as expected it is accurate for errors on the evidence $\log \mathcal{Z}$. See Appendix 3.B for a detailed discussion of the simulated weights method.

As the dimension d increases, Figure 3.6 shows parameter estimation errors decreasing and the evidence errors increasing (with a constant number of live points n). This

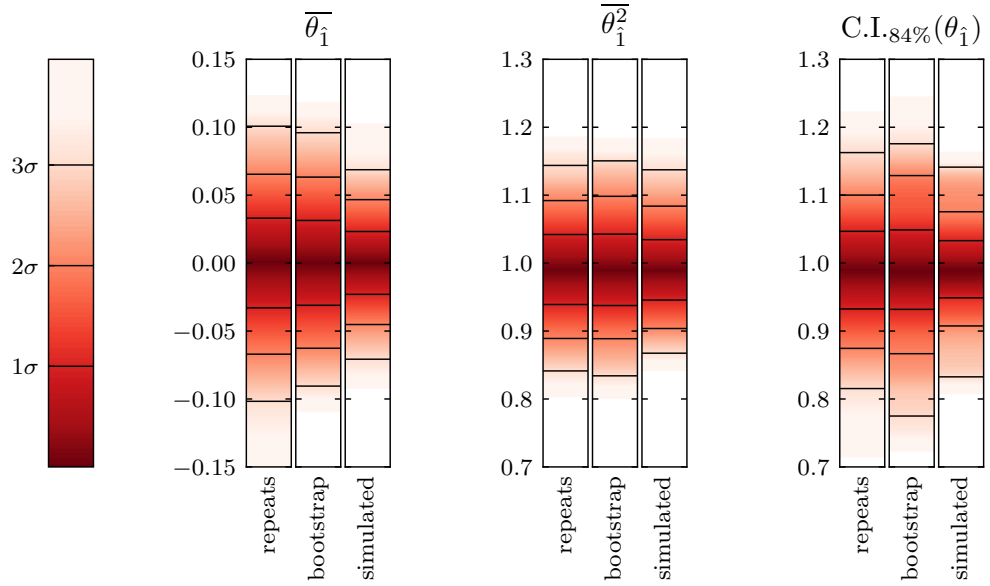


Figure 3.5: Estimated distributions of sampling errors for parameter estimation with a 3-dimensional Gaussian likelihood (3.7) for perfect nested sampling with $n = 200$. For each estimator the first plot uses values from 5,000 nested sampling runs; the second and third plot are calculated from a single nested sampling run and use 5,000 simulated weights and bootstrap replications. The bootstrap distributions are calculated using (3.19) and (3.20). The simulated weights and bootstrap values were adjusted by subtracting the difference between their run's expected value for each estimator and its analytical value to line up the distributions. The colour scale shows the fraction of the cumulative probability distribution lying between some region and the median.

effect is due to the posterior being contained in a smaller fraction of the prior volume in higher dimensions. In the spherically symmetric cases considered, the range of $\log X$ to be explored increases approximately linearly with the dimension d, as can be seen in Figure 3.3. With a constant number of live points, the number of samples is therefore also approximately proportional to d.

In parameter estimation from posterior samples only points' relative weights matter, so the increased number of samples in higher dimension problems typically increases accuracy as can be seen in Figures 3.6a and 3.6b. However for high dimensional Cauchy likelihoods (3.8) the posterior mass is spread over a wide range of $\log X$ values, so errors in the relative weights of points become large in high dimensions.[3]

[3]When the errors in points' relative weights become dominant the simulated weights method captures the majority of the sampling error, as can be seen for high dimensional Cauchy distributions in Figure 3.6b.

For $\log \mathcal{Z}$ the dominant error is in the absolute value of point weights, which is approximately proportional to the square root of the number of steps required to reach the posterior (Skilling, 2006). $\log \mathcal{Z}$ errors are therefore approximately proportional to $\sqrt{d}$ when n is constant, as can be seen in Figure 3.6c.

3.6 Application to existing nested sampling software

Nested sampling software can be easily modified to output information about the step at which dead points were sampled, which enables sampling error estimates using bootstrap resampling of threads. This has been incorporated into the most recent versions of `MultiNest` and `PolyChord`, and the nested sampling run division and sampling error estimates introduced in this chapter can be applied to results produced with these packages using `nestcheck`.

Sampling error estimates from our approach will be accurate provided the software is performing nested sampling approximately correctly. However such software can only approximately sample randomly from the prior within iso-likelihood contours — this may result in additional errors which are specific to a given implementation and which may not be captured by resampling threads. These additional errors are discussed in detail in Chapter 4.

We now demonstrate our method's application to nested sampling results produced with `PolyChord`.

3.6.1 Sampling errors on data fitting with `PolyChord`

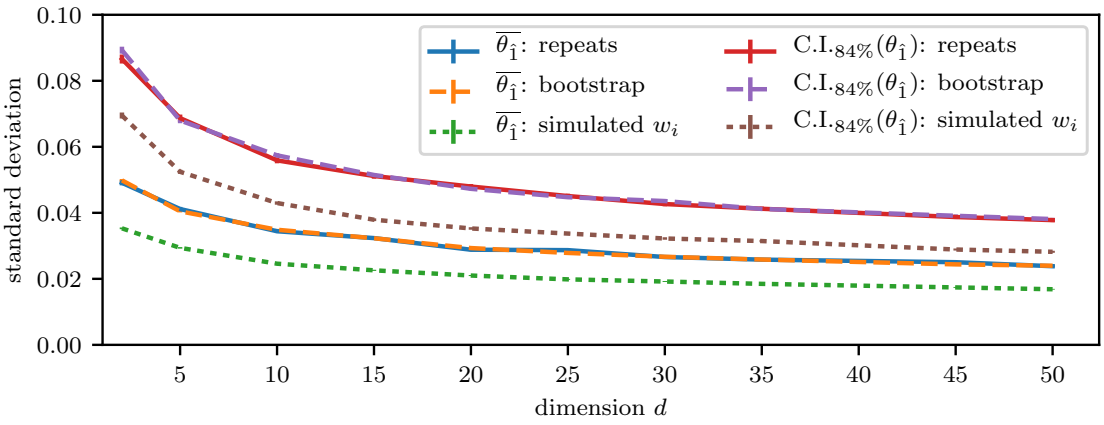
We fit a set of points $\mathcal{D} = \{x_i, y_i\}$ with normally distributed errors σ_y on the y values using a sinusoid

$$y(x) = A \sin(\omega x + \phi). \tag{3.21}$$

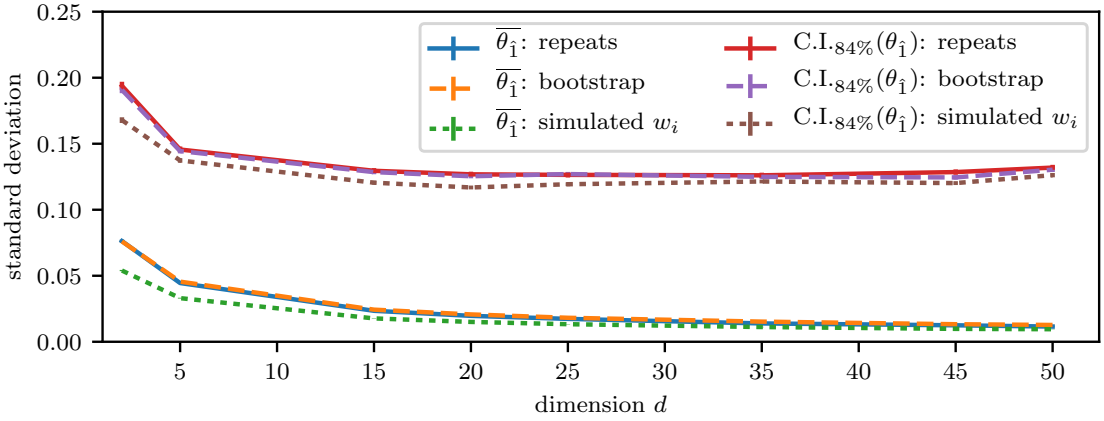
The likelihood is then

$$\mathcal{L}(\boldsymbol{\theta}) = \prod_i \frac{1}{\sqrt{2\pi {\sigma_y}^2}} \mathrm{e}^{-(y_i - y(x))^2 / 2\sigma_y^2}, \tag{3.22}$$

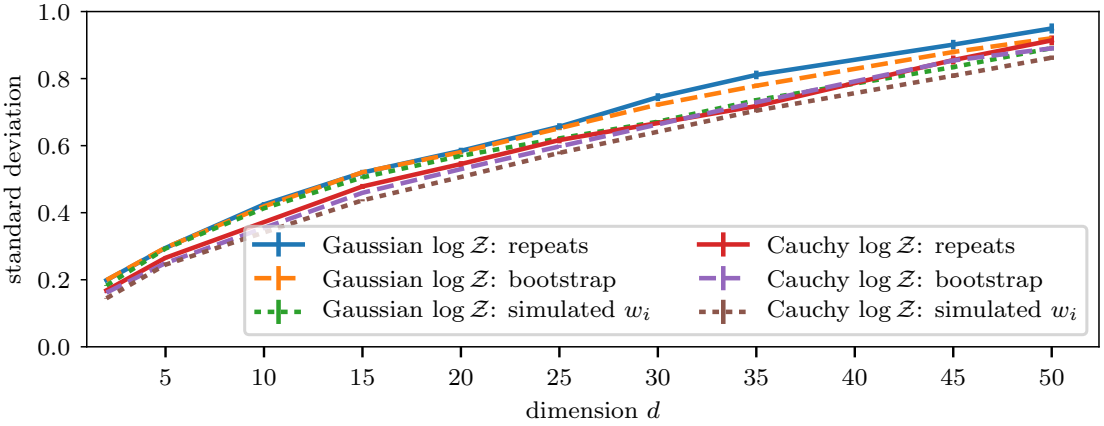
where $\boldsymbol{\theta} = (A, \omega, \phi)$ and we use a uniform prior for $A \in (0, 1)$, $\omega \in (0, 10)$ and $\phi \in (-\pi/2, \pi/2)$. Numerical tests use 40 data points sampled from $y(x) = \frac{1}{2}\sin(2\pi x)$ with Gaussian noise of size $\sigma_y = 0.2$ added to the y values; $y(x)$ and the data points are shown



(a) Parameter estimation sampling errors for Gaussian likelihoods (3.7).



(b) Parameter estimation sampling errors for Cauchy likelihoods (3.8).



(c) Evidence sampling errors for Gaussian (3.7) and Cauchy (3.8) likelihoods.

Figure 3.6: Sampling errors for likelihoods of different dimensions d; all use use d-dimensional Gaussian priors (3.9) and $n = 100$. Solid lines show the standard deviation of the results of 2,000 calculations. Dashed and dotted lines show the mean of 500 standard error estimates using the bootstrap and simulated weights methods respectively.

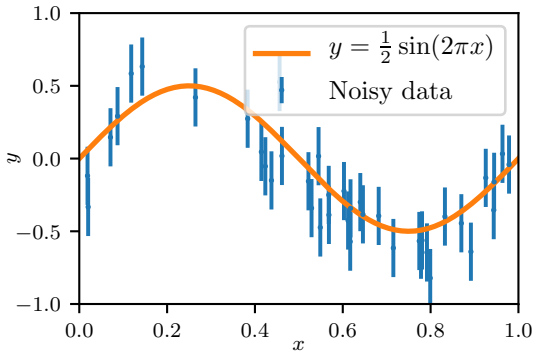


Figure 3.7: The true $y(x)$ function and the 40 data points used in numerical tests. Data has Gaussian noise of size $\sigma_y = 0.2$.

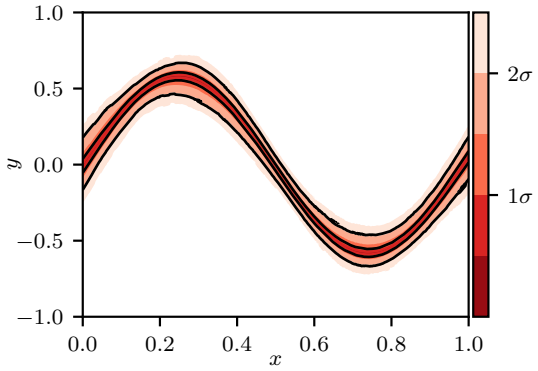


Figure 3.8: Posterior distribution on $y(x)$ given the data $\mathcal{D}$. The colour scale indicates credible intervals on $y(x)$.

in Figure 3.7 and the posterior distribution of $y(x)$ given the data is shown in Figure 3.8. Posterior distributions on A, ω and ϕ can be calculated with nested sampling — these are illustrated in Figure 3.9 along with example sampling errors.

Table 3.2 shows sampling errors from `PolyChord` with `num_repeats` $= 15$ — the default value for a 3-dimensional problem. As in perfect nested sampling, our bootstrap estimates of the standard error agree with the variation in results observed, and the observed coverage of credible intervals is close to their nominal coverage. This implies that `num_repeats` $= 15$ is sufficient for `PolyChord` to perform parameter estimation accurately in this case.

3.7 Application gravitational wave data analysis

The method for error estimation introduced in this chapter has a wide variety of possible applications in astronomy due to the widespread use of nested sampling parameter estimation in the field. As an example, we briefly describe our use of bootstrap resampling error estimates in Chua et al. (2018); more details can be found in the paper (on which the candidate is an author).

The first detection of gravitational waves by LIGO (LIGO Scientific Collaboration and Virgo Collaboration, 2016) heralded the dawn of a new era of astronomy, promising powerful new tests for probing fundamental physics. Future space-based detectors such as LISA (LISA Collaboration, 2017) will be able to detect binary mergers with extreme-mass-ratio inspirals, in which stellar-origin black holes or neutron stars merge

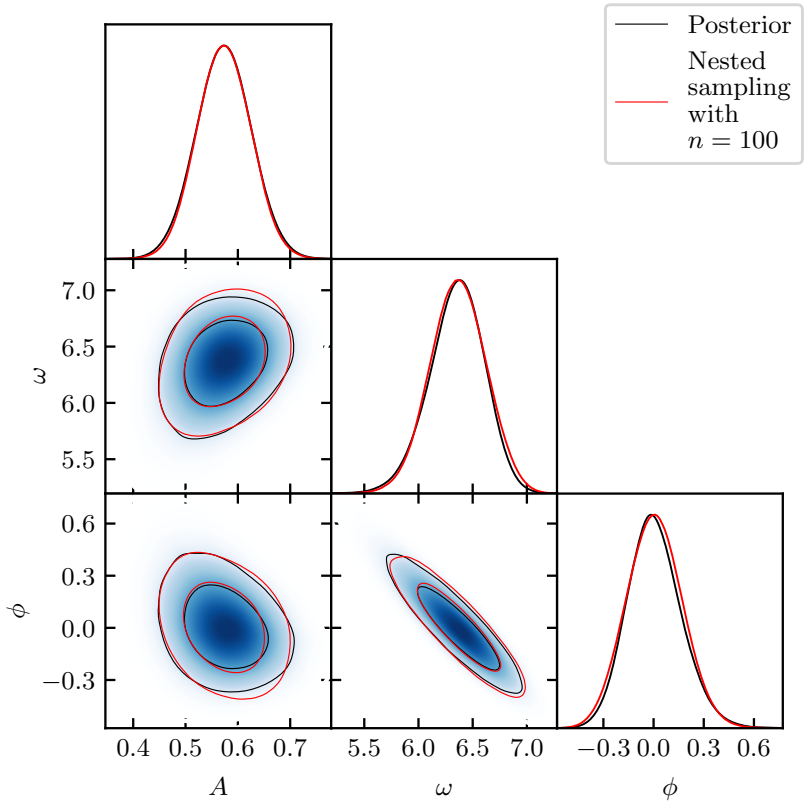


Figure 3.9: Posterior distributions and sampling errors from fitting a sinusoid to data (3.22) using `PolyChord`. The shading and black lines show an accurate calculation of posterior distribution and the 68% and 95% credible intervals from combining 1,000 nested sampling runs with $n = 100$. The red lines show the calculated posterior credible intervals for a single nested sampling run with $n = 100$, and differ from the true posterior due to sampling errors.

with massive black holes in galactic nuclei. As these events involve a large number of observable orbits during which the smaller object is in the strong gravitational field of the larger object, they can permit stringent tests of general relativity (Chua et al., 2018; Barack and Cutler, 2007; Gair et al., 2013).

Chua et al. (2018) trials testing for deviations from general relativity with a generalised extreme mass ratio inspiral waveform (Gair and Yunes, 2011) using the Bayesian null hypothesis test framework introduced for LIGO sources (Li et al., 2012). This involves adding N deformation parameters to the waveform from general relativity, which can be either be included or omitted (set to zero). The null hypothesis (general rela-

	A	ω	C.I.$_{84\%}(\omega)$
Repeated runs St.Dev.	$0.247(6) \cdot 10^{-2}$	0.012(0.3)	0.019(0.4)
BS St.Dev. / Repeats St.Dev.	0.98(2)	0.98(2)	0.97(2)
Sim St.Dev. / Repeats St.Dev.	0.71(2)	0.71(2)	0.73(2)
BS St.Dev. estimate variation	8.8(2)%	9.3(2)%	19.1(4)%
Sim estimate variation	6.4(1)%	6.6(1)%	21.5(5)%
BS C.I.$_{95\%}$	0.577(0.2)	6.370(0.8)	6.632(1)
BS Mean±1St.Dev. coverage	69.4%	69.3%	66.9%
BS C.I.$_{95\%}$ coverage	93.4%	95.0%	95.1%

Table 3.2: Sampling errors for the sinusoid fitting likelihood (3.22) using `PolyChord` with $n = 100$. The first row shows the standard deviation of 1,000 nested sampling calculations. The second and third rows show the mean of 1,000 error estimates from the bootstrap and simulated weights methods respectively as a ratio to the error observed from repeated calculations; 200 weight simulations and 200 bootstrap replications were used for each run. The fourth and fifth rows show the standard deviations of sampling error estimates for both methods as a percentage of the mean estimate. The sixth row shows the mean of 100 bootstrap estimates of the one-tailed 95% credible interval on the calculation result, each using 1,000 bootstrap replications. The final two rows show the empirical coverage of the bootstrap standard error and 95% credible interval from the 1,000 repeated calculations. Numbers in brackets show the error on the final digit.

tivity) model represents the case without any deformation parameters, and the $2^N - 1$ other models each involve some departure from general relativity.

One way to perform Bayesian model selection over the 2^N models is to calculate the Bayesian evidence for each independently (we term this the *vanilla method*). However, the vanilla method is extremely computationally expensive for this application. An alternative approach is the *adaptive method*, in which a single analysis is performed over a "meta-model" with an integer parameter which labels the different models. A sample's likelihood is evaluated by first selecting the model indicated by the integer parameter then evaluating the likelihood using that model and the remaining parameters (this approach is discussed in more detail in Section 6.3.4). Posterior odds ratios and Bayes factors for the different models can be calculated via parameter estimation over the integer parameter. Chua et al. (2018) used nested sampling and the adaptive method to test the models including deformation parameters against the general relativity model; as this is a parameter estimation problem, the associated errors are of the type discussed in this chapter.

Numerical calculations in Chua et al. (2018) used `PolyChord`, and uncertainties were evaluated using an earlier version of `nestcheck`; this proved useful as due to computational expense it was not possible to measure uncertainties accurately by repeating the analysis many times. The major result of the paper was the reduced uncertainty for a given computational cost possible using the adaptive method, with the reduction in uncertainty calculated using the methods introduced in this chapter.

3.8 Conclusion

Sampling errors in nested sampling parameter estimation arise principally from two sources: uncertain sample weights $w_i(\mathbf{t})$, and approximating the average of a function of parameters on each iso-likelihood contour $\tilde{f}(X_i)$ with a single sample $f(\boldsymbol{\theta}_i)$. The latter error is not present in evidence calculation and has been previously ignored. The added stochasticity from sampling each iso-likelihood contour makes nested sampling parameter estimation a 2-dimensional problem, with a dependence on both the distribution of posterior mass $\mathcal{L}(X)$ and the distribution of parameter values $P(f(\boldsymbol{\theta})|X)$ on each iso-likelihood contour. We proposed a new diagram for representing both aspects of the calculation, and presented it in Figures 3.2a to 3.2e.

Estimating sampling errors is vital for interpreting the results of a nested sampling calculation, as well as for allocating computational resources — for example by choosing an appropriate number of live points. However the previously available approach (the simulated weights method) underestimates sampling errors as it does not account for approximating $\tilde{f}(X_i)$ with a single sample $f(\boldsymbol{\theta}_i)$. We proposed a new method for estimating sampling errors using our new algorithm (Algorithm 2) for dividing a nested sampling run into single live point runs ("threads"), which can then be resampled with techniques such as the bootstrap. This works as the $\log X_i$ values of the dead points i from some nested sampling run with n live points form a Poisson process with rate n, meaning the $\log X_j$ values of the dead points j of a single thread are a Poisson process of rate 1.

Our method shows accurate estimation of sampling errors in parameter estimation in empirical tests, and compares favourably to the other methods discussed. Furthermore we have successfully applied the new method in the development of data analysis techniques for testing general relativity with gravitational waves. Our method can be

easily used together with existing nested sampling software, and will be reliable provided the implementation is performing the nested sampling algorithm accurately. The nested sampling run division and sampling error estimates introduced in this chapter are implemented in the `nestcheck` software package.

Appendix 3.A Relative contributions of different sources of parameter estimation sampling errors

The relative contributions of sampling errors from unknown prior weights of points $w_i(\mathbf{t})$ and from taking a single sample $\boldsymbol{\theta}_i$ on each iso-likelihood contour (discussed in Section 3.3) can be calculated by using exact values for weights $w_i(\mathbf{t})$ and replacing $f(\boldsymbol{\theta}_i)$ with $\tilde{f}(X_i)$. For a Gaussian likelihood (3.7) and Gaussian prior (3.9) both $w_i(\mathbf{t})$ and $\tilde{f}(X_i)$ can be calculated analytically for each $\boldsymbol{\theta}_i$ — sampling errors from calculations using this additional information are shown in Figure 3.10 and Table 3.3.

When calculating $\log \mathcal{Z}$, as expected, using exact weights $w_i(\mathbf{t})$ reduces uncertainty to the small trapezium rule error and using $\tilde{f}(X_i)$ has no effect. However for parameter estimation significant error remains when using exact $w_i(\mathbf{t})$ values.[4] The relative contribution to sampling errors from estimating weights statistically is greatest when $\tilde{f}(X)$ has a strong dependence on X over the interval in X containing the bulk of the posterior mass. In contrast when $f(\boldsymbol{\theta}) = \theta_{\hat{1}}$ then $\tilde{f}(X) = 0$ for all $\boldsymbol{\theta}$, and the analysis using $\tilde{f}(X_i)$ always gives the analytically correct answer of zero. In all cases, when both exact $w_i(\mathbf{t})$ and samples from $\tilde{f}(X_i)$ are used the sampling error is reduced to close to zero.

Appendix 3.B Analysis of the simulated weights method

The simulation method underestimates sampling errors in nested sampling parameter estimation, as shown by the numerical tests in Tables 3.1 and 3.2 and Figure 3.6. This is because it assumes that for each dead point $f(\boldsymbol{\theta}_i) \approx \tilde{f}(X_i)$, neglecting the sampling errors from taking a single sample on each iso-likelihood contour which are described in Section 3.3. However some of this error is captured because repeatedly simulating points'

[4] For $f(\boldsymbol{\theta}) = \theta_{\hat{1}}$ the error increases when exact $w_i(\mathbf{t})$ values are used. This is because the true weights are more variable than the expected ones and this reduces the information content (entropy) of the set of samples.

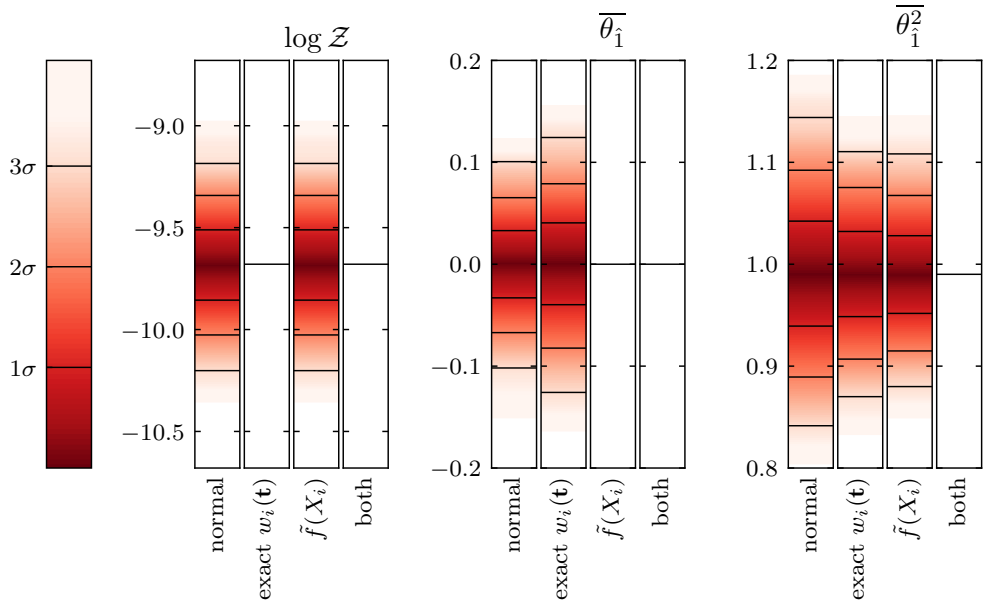


Figure 3.10: Sources of sampling error in perfect nested sampling with a 3-dimensional Gaussian likelihood (3.7), a Gaussian prior (3.9) and $n = 200$. Each plot shows the distribution of the results of 5,000 nested sampling calculations. For each estimator the first bar is from standard nested sampling, the second bar uses analytically calculated prior volumes for its sample weights $w_i(\mathbf{t})$ and the third bar uses $\tilde{f}(X_i)$ instead of $f(\boldsymbol{\theta})$ to calculate estimates. The fourth bar uses both analytical $w_i(\mathbf{t})$ and $\tilde{f}(X_i)$ values — the error in this case is very small and calculation results are all close to the analytic answer. The colour scale shows the fraction of the cumulative probability distribution lying between some region and the median.

	St.Dev.$[\{\log \mathcal{Z}\}]$	St.Dev.$\left[\left\{\overline{\theta_{\hat{1}}}\right\}\right]$	St.Dev.$\left[\left\{\overline{\theta_{\hat{1}}^2}\right\}\right]$
Normal runs	0.169(2)	0.033(3)	0.051(0.5)
Exact $w_i(\mathbf{t})$	$0.394(4) \cdot 10^{-5}$	0.040(4)	0.040(0.4)
Sampling $\tilde{f}(X_i)$	0.169(2)	0.000(0)	0.038(0.4)
Exact $w_i(\mathbf{t})$ and $\tilde{f}(X_i)$	$0.394(4) \cdot 10^{-5}$	0.000(0)	$0.330(3) \cdot 10^{-5}$

Table 3.3: The standard deviations of the sampling error distributions in Figure 3.10; numbers in brackets show the error on the final digit. For $f(\boldsymbol{\theta}) = \theta_{\hat{1}}$, $\tilde{f}(X) = 0$ for all $X(\boldsymbol{\theta})$ and so when $\tilde{f}(X_i)$ values are used every calculation gives $\overline{\theta_{\hat{1}}} = 0$ without any sampling error.

weights behaves like a resampling scheme, with similarities to the Bayesian bootstrap (Rubin, 1981). Resampling estimates the uncertainty on inferences from a set of samples by calculating its variation when data points are re-weighted, but the simulated weights method does so in a way that systematically underestimates sampling errors. This behavior has not been documented in the literature.

For example, consider the case $f(\boldsymbol{\theta}) = \theta_{\hat{1}}$ with a Gaussian likelihood (3.7) and Gaussian prior (3.9) — here $\tilde{f}(X) = 0$ for all $X(\boldsymbol{\theta})$. If $\tilde{f}(X_i)$ is used instead of $f(\boldsymbol{\theta}_i)$ there is no sampling error on estimates of $\overline{\theta_{\hat{1}}}$ regardless of any uncertainty in the weights of each point p_i, as can be seen in Figure 3.10 and Table 3.3. However the simulated weights method gives a non-zero estimate which on average differs from the sampling errors measured by repeated calculations by a factor of very close to $2^{-\frac{1}{2}} = 0.707$, as shown in the third row of Table 3.1.

Further numerical tests show that in special cases when $\tilde{f}(X)$ is the same at all X the ratio of sampling errors from the simulated weights method to the error observed in repeated calculations has a value close to $2^{-\frac{1}{2}}$. We give an analytical explanation for this result below. However we note that for practical problems $\tilde{f}(X)$ is *a priori* unknown and likely varies in X, meaning the true sampling error cannot be predicted by adjusting estimates from the simulated weights method.

3.B.1 Sampling error estimates for special cases when $\tilde{f}(X)$ is constant for all X

Variance of sampling error distribution

Nested sampling calculates the expected value of a function of parameters as $\sum_i p_i f(\boldsymbol{\theta}_i)$. Here the sampling error is the difference between the exact value of $\mathrm{E}[f(\boldsymbol{\theta})]$ from the posterior, and is distributed as

$$\text{sampling error} \;\sim\; P\left(\sum_{s\in\mathcal{S}} p_s f(\boldsymbol{\theta}_s) - \mathrm{E}[f(\boldsymbol{\theta})]\right). \tag{3.23}$$

The variance of this distribution provides a measure of the size of the sampling error. As the nested sampling estimator is unbiased, the variance of the sampling error distribution is equal to the variance of the results of repeated calculations:

$$\mathrm{Var}\left[\sum_i p_i(\mathbf{t}) f(\boldsymbol{\theta}_i)\right] = \sum_{i,j} \mathrm{Cov}\left[p_i(\mathbf{t}) f(\boldsymbol{\theta}_i), p_j(\mathbf{t}) f(\boldsymbol{\theta}_j)\right]. \tag{3.24}$$

Expanding and dropping the explicit dependence of p_i and f_i on $\mathbf{t}$ and $\boldsymbol{\theta}_i$ for brevity gives

$$\begin{aligned}\mathrm{Cov}\left[p_i f_i, p_j f_j\right] = \mathrm{E}[p_i]\mathrm{E}[p_j]\mathrm{Cov}[f_i, f_j] + \mathrm{E}[p_i]\mathrm{E}[f_j]\mathrm{Cov}[f_i, p_j] + \\ \mathrm{E}[f_i]\mathrm{E}[p_j]\mathrm{Cov}[p_i, f_j] + \mathrm{E}[f_i]\mathrm{E}[f_j]\mathrm{Cov}[p_i, p_j] + \\ \mathrm{E}[(\Delta p_i)(\Delta p_j)(\Delta f_i)(\Delta f_j)] + \mathrm{E}[p_i]\mathrm{E}[(\Delta f_i)(\Delta p_j)(\Delta f_j)] + \\ \mathrm{E}[f_i]\mathrm{E}[(\Delta p_i)(\Delta p_j)(\Delta f_j)] + \mathrm{E}[p_j]\mathrm{E}[(\Delta p_i)(\Delta f_i)(\Delta f_j)] + \\ \mathrm{E}[f_j]\mathrm{E}[(\Delta p_i)(\Delta f_i)(\Delta p_j)] - \mathrm{Cov}[p_i, f_i]\mathrm{Cov}[p_j, f_j],\end{aligned} \tag{3.25}$$

where $\Delta y \equiv y - \mathrm{E}[y]$.

Each f_i is an independent random variable from the distribution $P(f(\boldsymbol{\theta})|X_i)$, so the expectation of products of $\Delta p_i \Delta f_j$ are zero for all i, j. Furthermore expectation of products $\Delta f_i \Delta f_j$ and the covariance $\mathrm{Cov}[f_i, f_j]$ are zero for $i \neq j$.

The weights p_i have a dependence on X, but in the case $\tilde{f}(X_i)$ is the same for all X the covariance terms $\mathrm{Cov}[f_i, p_j]$ are also zero for all i, j. (3.25) therefore simplifies to

$$\sum_i \sum_j \mathrm{Cov}\left[p_i f_i, p_j f_j\right] = \sum_i \mathrm{Var}\left[p_i f_i\right] + \sum_{i \neq j, j} \left[\mathrm{E}[f_i]\mathrm{E}[f_j]\mathrm{Cov}[p_i, p_j]\right]. \tag{3.26}$$

Expanding the variance term on the right hand side when $\tilde{f}(X)$ is constant and f_i and p_i are therefore independent gives

$$\sum_{i,j} \mathrm{Cov}\left[p_i f_i, p_j f_j\right] = \sum_i \left[\mathrm{E}[{p_i}^2]\mathrm{Var}[f_i]\right] + \sum_{i,j} \left[\mathrm{E}[f_i]\mathrm{E}[f_j]\mathrm{Cov}[p_i, p_j]\right]. \tag{3.27}$$

Simulated weights method variance estimate

The simulated weights method corresponds to fixing the f_i values while retaining the stochastic dependence of p_i on $\mathbf{t}$. This means taking $\mathrm{E}[f_i]_{\mathrm{sim}} = f_i$, $\mathrm{Var}[f_i]_{\mathrm{sim}} = 0$, which combined with (3.27) gives

$$\mathrm{Var}_{\mathrm{simulated}} = \sum_{i,j} f_i f_j \mathrm{Cov}[p_i, p_j]. \tag{3.28}$$

Taking the expected values for f_i and f_j this becomes

$$\mathrm{E}[\mathrm{Var}_{\mathrm{simulated}}] = \sum_{i,j} \mathrm{E}[f_i]^2 \mathrm{Cov}[p_i, p_j] + \sum_i \left(\mathrm{E}[f_i^2] - \mathrm{E}[f_i]^2\right) \mathrm{Var}[p_i]. \tag{3.29}$$

Using that by definition $\sum_i p_i = 1$ so $\sum_{i,j} \mathrm{Cov}[p_i, p_j] = \mathrm{Var}[\sum_i p_i] = 0$,

$$\mathrm{E}[\mathrm{Var}_{\mathrm{simulated}}] = \sum_i \mathrm{Var}[f_i]\mathrm{Var}[p_i]. \tag{3.30}$$

In contrast the repeated runs method retains the sampling error of f_i on $\boldsymbol{\theta}_i$ and uses the expected values of the weight $\mathrm{E}[p_i]$. Hence for a large number of trials $\mathrm{E}[f_i]_{\mathrm{rep}} = \mathrm{E}[f_i] = \tilde{f}(X_i)$, $\mathrm{Var}[f_i]_{\mathrm{rep}} = \mathrm{Var}[f_i]$ for all i. Subbing into (3.27) gives

$$\mathrm{E}[\mathrm{Var}_{\mathrm{repeats}}] = \sum_i \mathrm{Var}[f_i]\mathrm{E}\left[{p_i}^2\right] + \sum_{i,j} \mathrm{E}[f_i]^2 \mathrm{Cov}[\mathrm{E}[p_i], E[p_j]]\,. \tag{3.31}$$

Using that $\sum_{i,j} \mathrm{Cov}[E[p_i], E[p_j]] = \mathrm{Var}[\sum_i E[p_i]] = 0$,

$$\mathrm{E}[\mathrm{Var}_{\mathrm{repeats}}] = \sum_i \mathrm{Var}[f_i]\mathrm{E}\left[{p_i}^2\right]. \tag{3.32}$$

Ratio of simulated weights and repeated runs variance estimates

Combining equations (3.30) and (3.32) gives the ratio of the simulated weights method and repeated runs variances as

$$\frac{\mathrm{E}[\mathrm{Var}_{\mathrm{simulated}}]}{\mathrm{E}[\mathrm{Var}_{\mathrm{repeats}}]} = \frac{\sum_i \mathrm{Var}[f_i]\mathrm{Var}[p_i]}{\sum_i \mathrm{Var}[f_i]\mathrm{E}\left[{p_i}^2\right]} = \frac{\sum_i \mathrm{Var}[f_i]\left(\mathrm{E}\left[{p_i}^2\right] - \mathrm{E}\left[p_i\right]^2\right)}{\sum_i \mathrm{Var}[f_i]E\left[{p_i}^2\right]}. \tag{3.33}$$

If $\mathrm{Var}[P(f(\boldsymbol{\theta})|X)]$ is the same for all X this simplifies to

$$\frac{\mathrm{E}[\mathrm{Var}_{\mathrm{simulated}}]}{\mathrm{E}[\mathrm{Var}_{\mathrm{repeats}}]} = \frac{\sum_i \mathrm{E}\left[{p_i}^2\right] - \mathrm{E}\left[p_i\right]^2}{\sum_i \mathrm{E}\left[{p_i}^2\right]}. \tag{3.34}$$

By definition the normalised weights $p_i \equiv \frac{w_i(\mathbf{t})}{\mathcal{Z}(\mathbf{t})}$ so

$$\mathrm{E}\left[p_i\right] = \mathrm{E}\left[w_i\right]\mathrm{E}\left[\mathcal{Z}^{-1}\right] + \mathrm{Cov}\left[w_i, \mathcal{Z}^{-1}\right], \tag{3.35}$$

$$\mathrm{E}\left[p_i^2\right] = \mathrm{E}\left[w_i^2\right]\mathrm{E}\left[\mathcal{Z}^{-2}\right] + \mathrm{Cov}\left[w_i^2, \mathcal{Z}^{-2}\right]. \tag{3.36}$$

Numerical results suggest that for a range of problems p_i and $\mathcal{Z}$ are approximately independent, in which case

$$\frac{\mathrm{E}[\mathrm{Var}_{\mathrm{simulated}}]}{\mathrm{E}[\mathrm{Var}_{\mathrm{repeats}}]} \approx \frac{\sum_i \mathrm{Var}[f_i]\left[\left(\mathrm{E}\left[{w_i}^2\right] - \mathrm{E}\left[w_i\right]^2\right) + \frac{\mathrm{Var}[\mathcal{Z}^{-1}]}{\mathrm{E}[\mathcal{Z}^{-2}]}\mathrm{E}\left[w_i^2\right]\right]}{\sum_i \mathrm{Var}[f_i]\mathrm{E}\left[{w_i}^2\right]}. \tag{3.37}$$

Typical problems with a large n often also have $\mathrm{Var}\left[\mathcal{Z}^{-1}\right] \ll \mathrm{E}\left[\mathcal{Z}^{-2}\right]$, in which case

$$\frac{\mathrm{E}[\mathrm{Var}_{\mathrm{simulated}}]}{\mathrm{E}[\mathrm{Var}_{\mathrm{repeats}}]} \approx \frac{\sum_i \mathrm{Var}[f_i]\left[\mathrm{E}\left[{w_i}^2\right] - \mathrm{E}\left[w_i\right]^2\right]}{\sum_i \mathrm{Var}[f_i]\mathrm{E}\left[{w_i}^2\right]}. \tag{3.38}$$

Keeton (2011) gives expressions for the weights as[5]

$$\mathrm{E}\left[w_i\right] = \mathrm{E}[\mathcal{L}_i]\frac{1}{n}\left(\frac{n}{n+1}\right)^i, \tag{3.39}$$

$$\mathrm{E}\left[w_i^2\right] = \mathrm{E}[\mathcal{L}_i]^2\frac{2}{n(n+1)}\left(\frac{n}{n+2}\right)^i. \tag{3.40}$$

For a general likelihood the summation in (3.38) cannot be found exactly. However one can estimate the ratio for each live point

$$\frac{\mathrm{E}\left[{w_i}^2\right] - \mathrm{E}\left[w_i\right]^2}{\mathrm{E}\left[{w_i}^2\right]} = \frac{\frac{2}{n(n+1)}\left(\frac{n}{n+2}\right)^i - \frac{1}{n^2}\left(\frac{n}{n+1}\right)^{2i}}{\frac{2}{n(n+1)}\left(\frac{n}{n+2}\right)^i} \tag{3.41}$$

$$= \frac{2 - \frac{n+1}{n}\left(1 - \frac{1}{(n+1)^2}\right)^i}{2} \tag{3.42}$$

$$\approx \frac{1}{2} \quad \text{when } n \gg 1 \text{ and } i \ll n^2. \tag{3.43}$$

This supports the observation that the ratio of simulated weights method estimates of the standard deviation of stochastic errors to measurements from repeated runs is close to $2^{-1/2}$ for special cases such as calculating the mean of a parameter for spherically symmetric likelihoods with spherically symmetric co-centred priors.

Ratio in the special case where $\mathcal{L}(X)$ and $P(f(\boldsymbol{\theta})|X)$ are constant for all X

If the likelihood $\mathcal{L}$ is constant[6] throughout the parameter space and $\mathrm{Var}[P(f(\boldsymbol{\theta})|X)]$ is the same for all X then the likelihood terms in the numerator and denominator of (3.38) cancel and the summation can be found exactly. Furthermore estimates of $\mathcal{Z}$ are very precise in this case as there is no stochastic variation in $\{\mathcal{L}_i\}$, justifying the approximation (3.38). In this case the ratio is

$$\frac{\mathrm{E}[\mathrm{Var}_{\text{simulated}}]}{\mathrm{E}[\mathrm{Var}_{\text{repeats}}]} \approx \frac{\sum_i \mathrm{E}\left[{w_i}^2\right] - \mathrm{E}\left[w_i\right]^2}{\sum_i \mathrm{E}\left[{w_i}^2\right]} \tag{3.44}$$

$$= \frac{\sum_i \left[\frac{1}{n}\left(\frac{n}{n+1}\right)^i + 1 - \frac{2}{n(n+1)}\left(\frac{n}{n+2}\right)^i\right]}{\sum_i \frac{2}{n(n+1)}\left(\frac{n}{n+2}\right)^i} \tag{3.45}$$

$$= \frac{1}{2 + \frac{1}{n}}, \tag{3.46}$$

[5]These formulae omit the trapezium rule and for brevity take $w_i(\mathbf{t}) = \mathcal{L}_i(X_{i-1} - X_i)$ — this approximation has little effect on the results.

[6]We assume $\mathcal{L}(X)$ has an infinitesimal slope to give direction to nested sampling's inward iteration.

	$\mathcal{Z}$	$\overline{\theta_{\hat{1}}}$	C.I.$_{84\%}(\overline{\theta_{\hat{1}}})$
Repeats St.Dev.	$0.111(1) \cdot 10^{-4}$	0.032(0.2)	0.055(0.4)
Split St.Dev. / Repeats St.Dev.	1.332(15)	1.012(8)	0.972(8)
BS St.Dev. / Repeats St.Dev.	1.009(8)	1.003(7)	1.008(8)
Split St.Dev. estimate variation	37.9(6)%	16.4(3)%	16.1(3)%
BS St.Dev. estimate variation	17.6(3)%	7.5(1)%	17.7(3)%

Table 3.4: Test of the split analysis method using perfect nested sampling with a 3-dimensional unit Gaussian likelihood (3.7), a Gaussian prior (3.9) and $n = 200$. The first row shows the standard deviation of results from 10,000 nested sampling calculations. The second row shows the mean estimate of sampling error standard deviation from 2,000 individual runs using the split method, breaking each run into 20 smaller runs with $n = 10$. The third row shows the mean of 2,000 bootstrap estimates of the sampling errors for comparison. The fourth and fifth row shows the standard deviation of error estimates from the split method and bootstrap method as a percentage of the mean estimate. Numbers in brackets show the error on the final digit.

where the final step sums the geometric series and neglects terms from the truncation of the sum due to termination of the nested sampling run.

Appendix 3.C Split runs method

Instead of spending all available computational resources on a single nested sampling run with n live points, one might consider performing N smaller runs with n/N live points and estimating the sampling error from the variation of the smaller runs — for example as $1/\sqrt{N}$ times their sample standard deviations. However this provides a limited number of sub-runs, and does not give accurate credible interval estimates. Furthermore while sampling errors in nested sampling are typically proportional to $1/\sqrt{n}$, this breaks down when the number of samples is small due to trapezium rule errors in sample weights which are $\mathcal{O}(1/n^2)$. As a result multiple runs are best analysed by combining them into a single run (Skilling, 2006).

Sampling error estimates from taking the standard deviation of the results of $N = 20$ sub-runs and multiplying by $1/\sqrt{N}$ are shown in Table 3.4. The split runs method is inaccurate for the approximately log-normally distributed sampling errors in $\mathcal{Z}$ as well as for credible intervals on distribution tails such as C.I.$_{84\%}(\overline{\theta_{\hat{1}}})$, as can be seen in the third row of Table 3.4.

Appendix 3.D Termination conditions

The sensitivity to termination conditions can be far higher for parameter estimation than for evidence calculation. This is both because parameter estimation can have much smaller sampling errors, and because the region close to the likelihood peak can have very high weight for some $f(\boldsymbol{\theta})$. For example for the Gaussian likelihood (3.7) an estimator such as $f(\boldsymbol{\theta}) = \overline{|\boldsymbol{\theta}|^{-1}}$ may show significant errors due to termination conditions which were perfectly adequate for calculating $\log \mathcal{Z}$. Numerical tests in this chapter use the termination conditions described by Handley et al. (2015b, Section 3.4), stopping when the estimated evidence contained in the live points is less than 10^{-4} times the evidence contained in dead points.

When splitting runs into their constituent threads (Section 3.4.1) then even in perfect nested sampling termination conditions must be chosen carefully to avoid causing differences between threads from different runs which terminate at different likelihoods. This typically happens when

1. termination conditions are worked out from the current set of dead points — e.g. estimating the evidence $\mathcal{Z}$ remaining as in Handley et al. (2015b, Section 3.4). This means some runs continue for longer than others;

2. the final point which violates the condition is kept. This means threads from small runs are much more likely to have final points far exceeding the termination condition than threads from large runs.

When comparing threads from different nested sampling runs, their equivalence can be maintained by using a termination condition which does not infer anything from the previous points, such as setting a fixed likelihood value $\mathcal{L}_{\text{term}}$ for termination and discarding any point that exceeds it. As we do not mix threads from different runs in our numerical tests we do not need this approach.

Appendix 3.E Additional numerical tests: 3-dimensional Cauchy likelihood

Table 3.5 shows numerical tests of sampling error estimates with a 3-dimensional Cauchy likelihood (3.8) with a Gaussian prior (3.9). As in the 3-dimensional Gaussian case

	$\overline{\theta_{\hat{1}}}$	$\overline{\theta_{\hat{1}}^2}$	C.I.$_{84\%}(\overline{\theta_{\hat{1}}})$
Repeated runs St.Dev.	0.044(0.3)	0.573(4)	0.119(0.8)
BS St.Dev. / Repeats St.Dev.	1.005(7)	1.003(8)	1.002(8)
Sim St.Dev. / Repeats St.Dev.	0.717(5)	0.994(8)	0.926(7)
BS St.Dev. estimate variation	9.3(1)%	12.7(2)%	16.9(3)%
Sim estimate variation	8.0(1)%	12.0(2)%	17.4(3)%
BS C.I.$_{95\%}$	0.072(5)	6.70(7)	1.69(2)
BS Mean$\pm$1St.Dev. coverage	68.6%	68.8%	68.7%
BS C.I.$_{95\%}$ coverage	95.1%	92.1%	92.1%

Table 3.5: Sampling errors for a 3-dimensional Cauchy likelihood (3.8), a Gaussian prior (3.9) and $n = 200$. The first row shows the standard deviation of 10,000 nested sampling calculations. The second and third rows show the mean of 2,000 error estimates from the bootstrap and simulated weights methods respectively as a ratio to the error observed from repeated calculations; 200 weight simulations and 200 bootstrap replications were used for each run. The fourth and fifth rows show the standard deviations of sampling error estimates for both methods as a percentage of the mean estimate. The sixth row shows the mean of 100 bootstrap estimates of the one-tailed 95% credible interval on the calculation result given the sampling error, each using 1,000 bootstrap replications. The final two rows show the empirical coverage of the bootstrap standard error and 95% credible interval from the 10,000 repeated calculations. Numbers in brackets show the error on the final digit.

shown in Table 3.1, the mean estimates of sampling errors from our bootstrap method are very close to measurements of sampling errors from repeated calculations — this can be seen in the second row of Table 3.5. Again the empirical coverage rates for bootstrap credible intervals are close to their nominal values, as shown in the final two rows.

Chapter 4

Diagnostic tests for nested sampling calculations

This chapter introduces new diagnostic tests to assess the reliability both of parameter estimation and evidence calculations using nested sampling software, and demonstrates them empirically. We present two new diagnostic plots for nested sampling, and give practical advice for nested sampling software users. The material presented is an edited version of Higson et al. (2019b).

4.1 Introduction

Methods for numerically estimating the uncertainty in nested sampling results due to the stochasticity of the nested sampling algorithm are now available for both evidence calculations (see Skilling, 2006; Keeton, 2011) and parameter estimation (see Chapter 3). However, all of these techniques assume that the nested sampling algorithm was executed perfectly — which requires sampling randomly from the prior within a hard likelihood constraint. This can only be done exactly in special cases, such as for spherically symmetric calculations using `perfectns` (Higson, 2018c). Nested sampling software used for practical problems can only perform such sampling approximately and as a result may produce additional errors — for example due to correlations between samples, or due to sampling from only part of the prior volume contained within a likelihood constraint. We term these additional errors *implementation-specific effects* to distinguish them from the intrinsic stochasticity of the nested sampling algorithm.

Diagnosing whether significant implementation-specific effects are present is of great practical importance for researchers as they can cause large uncertainty in results and lead to potentially incorrect conclusions — such as, for example, if the calculation misses a significant mode[1] in a multimodal posterior. Conversely, if implementation-specific effects are shown to be negligible, users can simply increase the number of live points for more accurate results and can confidently use standard techniques to estimate numerical uncertainty from the nested sampling algorithm.

Typically software has settings which the user can adjust to reduce implementation-specific effects at the cost of increased computation, such as `PolyChord`'s `num_repeats` and `MultiNest`'s `efr` (see Section 4.6 for more details). Assessing if the software is able to explore the posterior reliably is therefore particularly useful when taking significantly more samples is computationally costly, as is often the case for high-dimensional problems. In our experience, software users typically try to check their results by running a calculation several times and qualitatively assessing if the posterior distributions look similar in each case. However this is not very reliable and does not differentiate between implementation-specific effects and the expected variation from the inherent stochasticity of the nested sampling algorithm.

We are not aware of any diagnostic tests in the literature for checking calculation results for practical problems for implementation-specific effects, although Buchner (2016) proposes a diagnostic for evidence calculations which uses analytically solvable test problems. In contrast Markov chain Monte Carlo (MCMC)-based methods, which do not require sampling within a hard likelihood constraint, have an extensive literature on diagnostics for practical problems (see for example Cowles and Carlin, 1996; Hogg and Foreman-Mackey, 2018).

This chapter introduces new heuristic tests and diagrams to check the reliability of nested sampling results for practical problems, and to determine if the software settings should be changed. It is also intended to serve as a practical guide for nested sampling practitioners based on our experience using nested sampling software. We begin discussing the challenges of detecting implementation-specific effects in Section 4.2. We

[1]Here we refer to cases where the software does not detect the mode and, as a result, samples are not drawn from the entire prior volume within specified likelihood constraints. Another less common problem is that, if the number of live points is very low, a given run might not contain a single sample within a particular mode even when the nested sampling algorithm is performed perfectly; this is not an implementation-specific effect according to our definition.

then introduce our new diagnostic tests:

- Section 4.3 discusses diagnostic plots and presents two new diagrams for nested sampling (illustrated in Figures 4.2 to 4.4);
- Section 4.4 describes how the implementation-specific effects can be measured from a number of nested sampling runs;
- Section 4.5 introduces diagnostic tests which can be applied to pairs of nested sampling runs and are useful when few runs are available.

We empirically test the effects of changing nested sampling software settings and the dimension of the problem on both implementation-specific effects and total calculation errors using `PolyChord` in Section 4.6, and summarise our practical advice for software users in Section 4.6.5. Finally in Section 4.7 we apply our methods to astronomical data from the *Planck* survey. Our diagnostic tests and diagrams are implemented in `nestcheck` (Higson, 2018b); an open-source Python package for analysing nested sampling calculations, which is compatible with output from `MultiNest` and `PolyChord`.

4.2 Measuring implementation-specific effects

This chapter is concerned with developing practical diagnostics for assessing whether nested sampling calculation results contain implementation-specific effects due to imperfect execution of the nested sampling algorithm. It is important to emphasis that diagnosing such effects without additional information about the likelihood and prior is very challenging problem, and it is impossible to conclude *a priori* with certainty that they are not present. For example, one cannot eliminate the possibility of missing an extremely narrow mode for a general posterior without an exhaustive search of the parameter space (Wolpert and Macready, 1997). Hogg and Foreman-Mackey (2018, Section 5) provide an interesting and analogous discussion of the similarly heuristic nature of MCMC convergence tests. In addition, nested sampling's iteration towards successively higher likelihoods means it never reaches a steady state — so heuristics based on autocorrelation of samples (which are used in testing for MCMC convergence) cannot be applied.

The main idea behind the diagnostic tests we present is to assess if the variation of the results of different nested sampling runs is consistent with the statistical properties expected of nested sampling without implementation-specific effects. Consequently, these diagnostics require multiple nested sampling runs. A limitation of this approach is that a systematic bias in the calculation results will lead to the implementation-specific effects being underestimated, although they are still likely to be detectable. Such cases have been observed in the literature for evidence calculations with challenging posteriors (see for example Beaujean and Caldwell, 2013); we discuss systematic bias in detail in Section 4.6.3. Furthermore our diagnostics are unable to detect implementation-specific effects which do not change the variation of the runs, although we have not come across such a case in practice. A theoretical example would be if every run available missed a significant mode while exploring all the rest of the parameter space correctly.

4.2.1 Test problems

We now introduce two test problems, which we will use to demonstrate the diagnostic tests presented in the following sections.

As an example of a simple likelihood, we consider a simple d-dimensional Gaussian with $\sigma = 1$ centred on the origin

$$\mathcal{L}(\boldsymbol{\theta}) = (2\pi)^{-d/2}\mathrm{e}^{-|\boldsymbol{\theta}|^2/2}. \tag{3.7 revisited}$$

We also use the challenging LogGamma-Gaussian mixture model likelihood introduced by Beaujean and Caldwell (2013), which was designed to represent a particle physics problem involving heavy-tailed distributions and several distinct modes. In this case $\mathcal{L}(\boldsymbol{\theta}) = \prod_{i=1}^{d} \mathcal{L}(\theta_{\hat{i}})$ with

$$\begin{aligned}
\mathcal{L}(\theta_{\hat{1}}) &= \frac{1}{2}\mathrm{LogGamma}(\theta_{\hat{1}} - 10|1,1) + \frac{1}{2}\mathrm{LogGamma}(\theta_{\hat{1}} + 10|1,1),\\
\mathcal{L}(\theta_{\hat{2}}) &= \frac{1}{2}\mathrm{Normal}(\theta_{\hat{2}} - 10|0,1) + \frac{1}{2}\mathrm{LogGamma}(\theta_{\hat{2}} + 10|0,1),\\
&\text{and, if } d > 2,\\
\mathcal{L}(\theta_{\hat{i}}) &= \begin{cases} \mathrm{LogGamma}(\theta_{\hat{i}}|1,1) & \text{for } 3 \leq i \leq \frac{d+2}{2},\\ \mathrm{Normal}(\theta_{\hat{i}}|0,1) & \text{for } \frac{d+2}{2} \leq i \leq d. \end{cases}
\end{aligned} \tag{4.1}$$

Here the number of dimensions d is even and the LogGamma distribution is

$$\mathrm{LogGamma}(x|\alpha,\beta) = \frac{\mathrm{e}^{\beta x}\mathrm{e}^{-\mathrm{e}^x/\alpha}}{\alpha^\beta \Gamma(\beta)}, \tag{4.2}$$

where Γ denotes the gamma function.

Our numerical tests all use uniform priors $\in [-30, 30]$ for each parameter. As (4.1) and (3.7) are both normalised to 1 and there is negligible posterior mass outside the prior, in both cases the evidence is almost exactly equal to the normalisation constant on the uniform prior — i.e.

$$\mathcal{Z}_{\text{true}} = 60^{-d}. \tag{4.3}$$

4.3 Diagnostic plots

Before discussing quantitative diagnostics in Sections 4.4 and 4.5, we first introduce some diagnostic plots which illustrate nested sampling and its associated errors. It is good practice for users of sampling software to represent their results visually, in order to assess if they are reasonable given background knowledge about the problem. Many software packages exist for plotting 1- and 2-dimensional marginalised distributions from weighted samples using kernel density estimation. As an example, Figure 4.1 shows posterior distributions for the LogGamma mixture likelihood (4.1); this was made using `getdist` with a zero-centred Gaussian kernel and the default settings.

While plots like Figure 4.1 are useful, it is unclear to what extent the differences between the two nested sampling runs are due to implementation-specific effects or merely what is expected from the stochasticity of the nested sampling algorithm. Furthermore, these plots do not illustrate the distinctive manner in which nested sampling iterates towards higher likelihoods. We therefore propose two additional diagnostic plots in Sections 4.3.1 and 4.3.2, which can be calculated from nested sampling runs to show this extra information. These are focused on distributions of parameters and so do not directly assess evidence calculations, but any significant inconsistencies in sample allocations observed between runs may also impact evidence estimates.

4.3.1 Plotting the uncertainty on posterior distributions

The uncertainty on the posterior distributions due to nested sampling stochasticity can be estimated from a run by creating bootstrap resamples of the run using the procedure described in Section 3.4.2. This uncertainty can be visually represented by plotting the distribution of the posteriors obtained from each resample (which is a nested sampling run) to give an *uncertainty distribution on the posterior distribution*. Such plots can

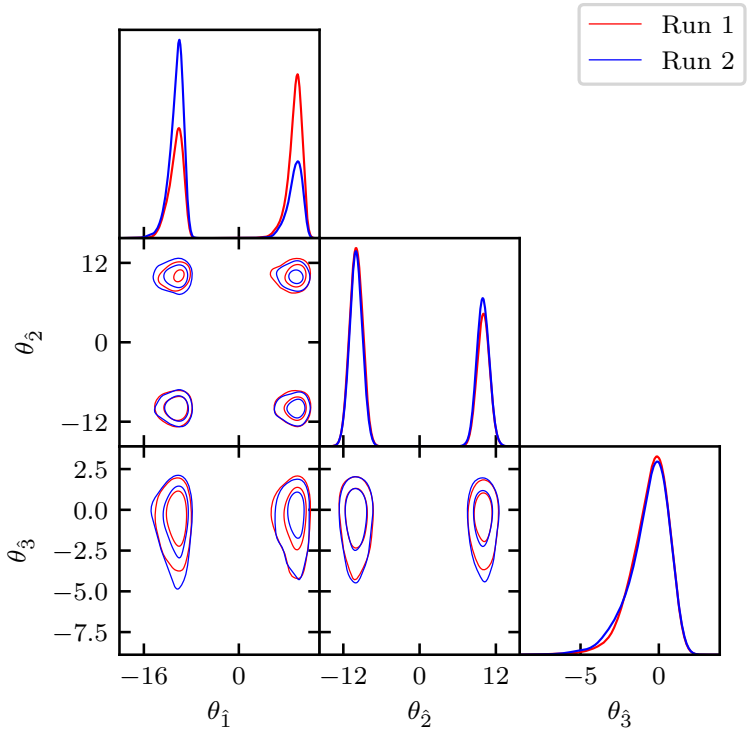


Figure 4.1: Triangle plot of the posterior distributions for two nested sampling runs (red and blue lines), calculated using the 10-dimensional LogGamma mixture likelihood (4.1) and a uniform prior. The on-diagonal plots show 1-dimensional marginalised posterior distributions on the first three parameters, and the remaining plots show calculated 2-dimensional 68% and 95% credible intervals on the joint posterior distribution. The results for the two runs differ due to errors from both the intrinsic stochasticity of the nested sampling algorithm and implementation-specific effects. Each nested sampling run has 250 live points, and uses the `PolyChord` setting `num_repeats` = 20 — this low setting is deliberately chosen to illustrate large implementation-specific effects.

be used for assessing if the calculation error is sufficiently small for the given use case, and are illustrated in Figure 4.2. If they are of interest, the posterior distributions of functions of parameters can also be plotted; Figures 4.2a and 4.2b both show the radial coordinate $|\boldsymbol{\theta}| = (\sum_i \theta_{\hat{i}}^2)^{1/2}$. The coloured contours are plotted using the `fgivenx` package (Handley, 2018).[2]

Plotting results from multiple runs on the same axis allows visual assessment of whether implementation-specific effects are present. If posterior distributions differ by more than would be expected from their bootstrap sampling error distribution, then implementation-specific effects are likely to be the cause. For example the top left panel of Figure 4.2b, in which the coloured distributions are clearly separated, suggests large implementation-specific effects are present in this case with the settings used. Figure 4.2 deliberately uses low values for the `PolyChord` `num_repeats` and number of live points settings to illustrate implementation-specific effects; these effect can be reduced with a more appropriate choice of settings (discussed in Section 4.6).

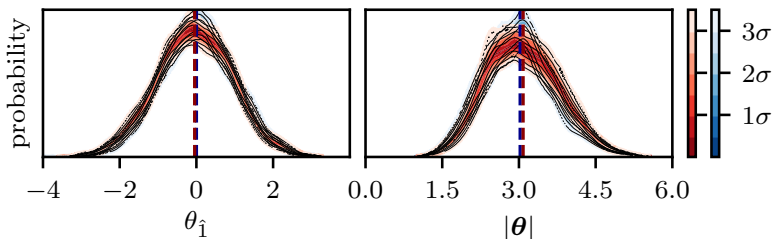
4.3.2 Plotting distributions of samples in $\log X$

We now propose a diagram to illustrate the distinctive manner in which a nested sampling run progresses by sampling from the prior with successively higher likelihood constraints, based on the plots shown in Figure 3.2. This involves plotting sample parameters and weights against the fraction of the prior volume remaining, X, which is defined in (2.16). A log scale is used as the shrinkage in X at each step is exponential.

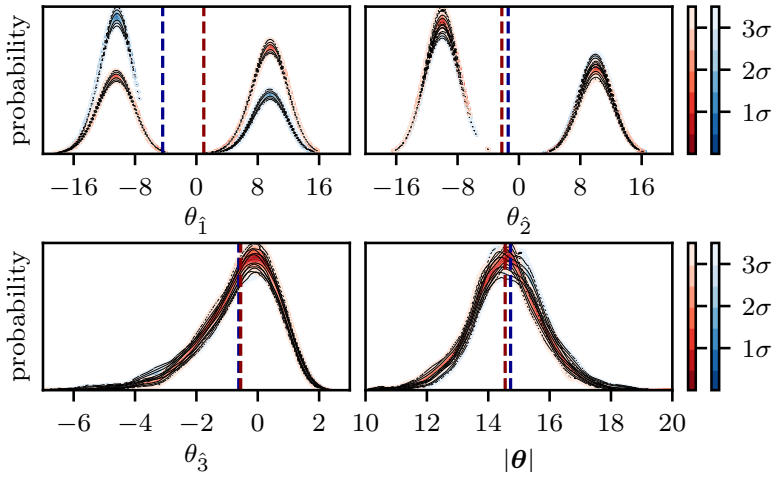
In each plot the top right panel shows the relative posterior mass $\mathcal{L}(X)X$ (i.e. the weight assigned to samples in that $\log X$ region) on a relative scale; this is similar to Figure 2.1. The $\log X$ co-ordinates of the samples are estimated statistically, with their uncertainty distribution displayed using coloured contours. Each subsequent row represents a parameter or function of parameters, with the right panel showing the parameter value of each sample on the same $\log X$ scale.[3] The left panel is the same

[2]When calculating plots like those in Figure 4.2, the posterior distribution for each bootstrap replication must be calculated from the weighted samples without reducing them to evenly weighted samples in a stochastic manner — such as by including each sample with probability proportional to its weight — as this adds extra variation. `nestcheck` contains an implementation of 1-dimensional kernel density estimation which takes sample weights as an argument, and does not require conversion to evenly weighted samples.

[3]The scatter plots in the right column of Figures 4.3 and 4.4 can be replaced with a colour plot of the estimated distribution of values at each $\log X$ using kernel density estimation (similar to the colour



(a) Posterior distributions of the first parameter and the radial coordinate $|\boldsymbol{\theta}|$ for a 10-dimensional Gaussian likelihood (3.7).



(b) Posterior distributions of the first 3 parameters and $|\boldsymbol{\theta}|$ for a 10-dimensional LogGamma mixture likelihood (4.1). The nested sampling runs are the same ones used in Figure 4.1 with the corresponding colours.

Figure 4.2: Diagrams of posterior distributions for two nested sampling runs (red and blue), showing the uncertainty due to the stochasticity of the nested sampling algorithm. Each run uses 250 live points, and has `num_repeats` = 20 deliberately set to a low value to illustrate implementation-specific effects. The coloured contours show iso-probability credible intervals on the marginalised posterior probability density function at each parameter value. The dashed dark blue and dark red lines show the estimated posterior means of each parameter for the blue and red runs respectively.

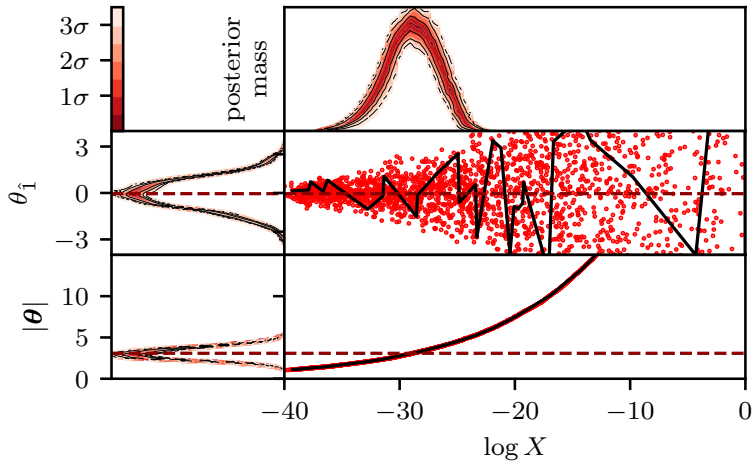


Figure 4.3: Diagram of samples' distributions in $\log X$ for a single run with a 10-dimensional Gaussian likelihood (3.7). The top right panel shows the relative posterior mass (total weight assigned to all samples in that region) as a function of $\log X$. The next two rows show the first parameter and the radial coordinate $|\boldsymbol{\theta}|$; for each the right panel plots its sampled values against $\log X$ and the left panel shows its posterior distribution in the same way as Figures 4.2a and 4.2b. The coloured contours show iso-probability credible intervals on the marginalised posterior probability density function at each parameter or $\log X$ value. The nested sampling run shown uses 250 live points and `num_repeats` $= 20$. The solid black line shows the evolution of an individual thread (chosen at random). The estimated mean value of the posterior distribution for each row is marked with a dashed line.

as the plots in the previous section (Figures 4.2a and 4.2b), and shows the posterior distribution on the parameter values on a shared scale with the right panel (including the uncertainty due to the stochasticity of the nested sampling algorithm).

Our proposed diagram is illustrated in Figures 4.3 and 4.4. The lower limit of the $\log X$ axis is chosen to include all points with non-negligible posterior mass, and the upper limit is set to 0 (the start of the nested sampling run). The y-axis limits of the plots in the right column are simply chosen to include all samples with non-negligible posterior weight, or which are otherwise of interest.

In addition, the evolution of individual threads can be traced by drawing lines linking their constituent points.[4] This shares similarities with MCMC trace plots but, unlike

distributions shown in Figure 3.2). However doing this accurately is computationally challenging and requires a lot of samples, so simple scatter plots are typically more convenient for checking calculation results.

[4]Plots which trace individual threads in $\log X$ are also produced by the `dynesty` dynamic nested sampling package. See `https://github.com/joshspeagle/dynesty` for more information.

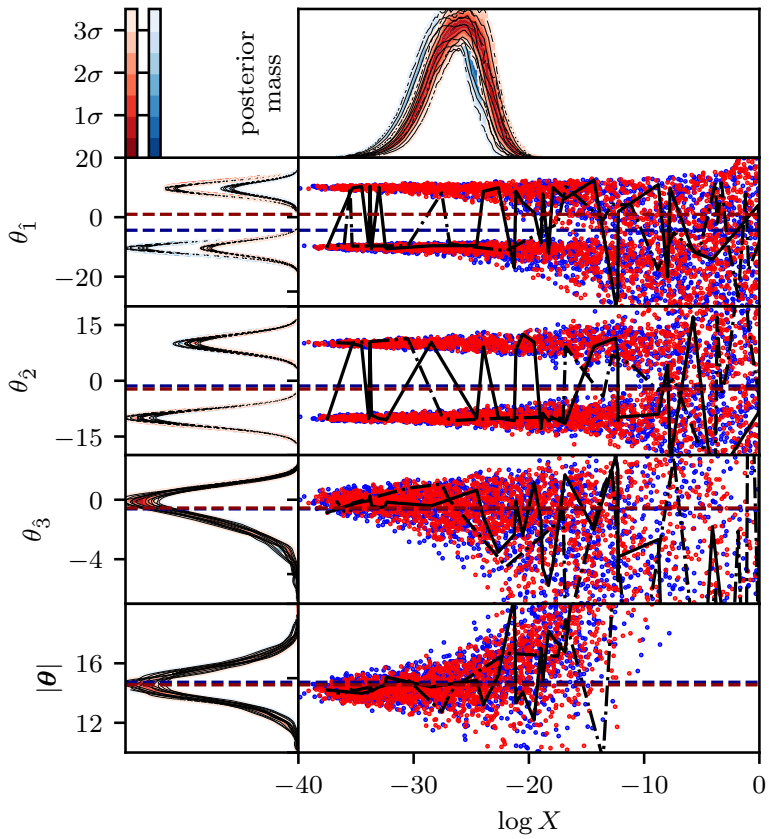


Figure 4.4: Diagram of samples' distributions in $\log X$ for two nested sampling runs from a 10-dimensional LogGamma mixture likelihood (4.1). The two runs (shown in red and blue) are the same ones used for Figure 4.1 and Figure 4.2b; each uses 250 live points and `num_repeats` $= 20$. The top right panel shows the relative posterior mass (total weight assigned to all samples in that region) as a function of $\log X$. The next four rows show the first 3 parameters and the radial co-ordinate $|\boldsymbol{\theta}|$; for each the right panel plots its sampled values against $\log X$ and the left panel shows its posterior distribution in the same way as Figures 4.2a and 4.2b. The coloured contours show iso-probability credible intervals on the marginalised posterior probability density function at each parameter or $\log X$ value. In each row, the estimated posterior means for the blue and red runs are shown with dashed dark blue and dark red lines. The solid and dot dash black lines show the evolution of an individual thread chosen at random from the red and blue runs respectively.

for a converged MCMC chain, the distribution of parameters changes as the algorithm iterates over different $\log X$ values. Furthermore, as the algorithm progresses towards lower values of $\log X$ it moves from right to left in the diagram; in MCMC trace plots, chains typically move from left to right.

Figures 4.3 and 4.4 are useful for visualising the nested sampling process and parts of the posterior such as degeneracies and modes with which nested sampling software may struggle. Furthermore if additional information about the posteriors is available, such as that they should have certain symmetries or be unimodal, this type of diagram can be useful in working out where the sampler is not behaving as expected. For example Figure 4.4 clearly shows the multi-modality of the LogGamma mixture likelihood, as well as giving an indication of when in the nested sampling process the modes separate. In addition the bottom right panel of Figure 4.3 shows that the radial coordinate $|\boldsymbol{\theta}|$ has negligible spread at any given $\log X$ value in this case; this is due to the likelihood and prior's spherical symmetry.

Furthermore, multiple nested sampling runs can be added to the same axis — as shown in Figure 4.4. This allows comparison of where runs differ; for example one may be able to see on the plot that one of the runs had missed a mode which the other run found (although in Figure 4.4 the samples from the two runs overlap). One can also see from Figure 4.4 that the two runs agree closely on the relative weights assigned at different $\log X$ values (top panel), meaning that the difference between the posterior distributions (left panels) is due to the parameter values sampled in each $\log X$ region rather than the distribution of posterior mass.

4.4 Estimating implementation-specific effects

Following the diagnostics plots of the previous section, the remainder of this chapter discusses quantitatively measuring implementation-specific effects. The total error on nested sampling calculations can be estimated by measuring the variation of results when a calculation is repeated multiple times, as this includes both implementation-specific effects and the intrinsic stochasticity of the algorithm. This provides a lower bound on the total error, but will underestimate it in the case that implementation-specific effects cause calculation results to be systematically biased.

While the nature of implementation-specific effects depends on the specific software

used, they are very likely to be uncorrelated with the errors from the stochasticity of the nested sampling algorithm — which can be calculated using the bootstrap resampling approach. Assuming that they are indeed uncorrelated, the variance in posterior inferences (such as the calculated values of parameter means or the Bayesian evidence) due to implementation-specific effects σ^2_{imp} is related to the variance estimated from bootstrap resampling σ^2_{bs} and the variance of calculation results σ^2_{values} by the standard relation for the sum of the variances of uncorrelated random variables (the Bienaymé formula)

$$\sigma^2_{\text{values}} = \sigma^2_{\text{bs}} + \sigma^2_{\text{imp}}. \tag{4.4}$$

Using this result, we propose calculating the standard deviation of the uncertainty distribution due to implementation-specific effects σ_{imp} as

$$\sigma_{\text{imp}} = \begin{cases} \sqrt{\sigma^2_{\text{values}} - \sigma^2_{\text{bs}}} & \text{if } \sigma^2_{\text{values}} > \sigma^2_{\text{bs}}, \\ 0 & \text{otherwise.} \end{cases} \tag{4.5}$$

If a number of nested sampling runs are available, the implementation-specific effects on calculations of scalar quantities such as the mean and median of parameters can be calculated directly from (4.5) and compared to the variation of results. One can also estimate the fraction of the observed variation which is due to implementation-specific effects $\sigma_{\text{imp}}/\sigma_{\text{values}}$ — when implementation-specific effects are large this is easy to measure accurately as the variation of results is much greater than the bootstrap error estimates and

$$\frac{\sigma_{\text{imp}}}{\sigma_{\text{values}}} = \frac{\sqrt{\sigma^2_{\text{values}} - \sigma^2_{\text{bs}}}}{\sigma_{\text{values}}} = 1 - \frac{\sigma_{\text{bs}}}{2\sigma_{\text{values}}} + \mathcal{O}\left(\frac{\sigma^2_{\text{bs}}}{\sigma^2_{\text{values}}}\right). \tag{4.6}$$

The number of runs required to estimate σ_{imp} is primarily determined by the accuracy of the sample standard deviation σ_{values}. Ahn and Fessler (2003) give a formula for the fractional uncertainty of the sample standard deviation as a function of the number of data points; for computationally expensive problems in our research, we typically use ~ 10 runs to estimate σ_{imp}. In practice σ_{bs} makes a negligible contribution to the uncertainty on σ_{imp}; it can be estimated accurately from a single run, and the accuracy can be further improved by averaging estimates from all the runs available.

Figure 4.5 shows the ratio of the inferred implementation error to the total variation of results for 100 nested sampling runs using 10-dimensional Gaussian (3.7) and

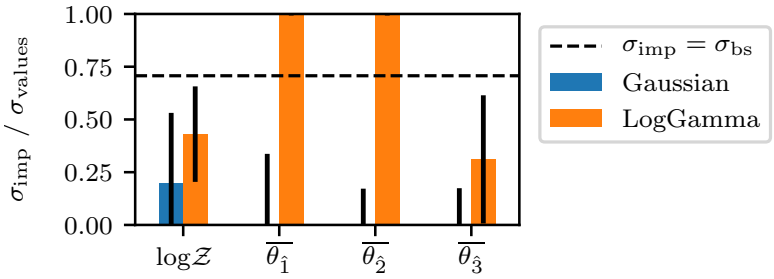


Figure 4.5: Ratios of estimated implementation-specific effects (4.5) to variation of results for 10-dimensional Gaussian (3.7) and LogGamma mixture (4.1) likelihoods. The dashed horizontal line at $\sigma_{\text{imp}}/\sigma_{\text{values}} = \frac{1}{\sqrt{2}}$ shows the level where implementation-specific effects and the stochasticity of the nested sampling algorithm make equal contributions to the total error; ratios above this value imply the majority of the error is due to implementation-specific effects. Each bar is calculated using 100 `PolyChord` runs, each with 250 live points and `num_repeats` = 50. Results are shown for the log-evidence, the mean of the two parameters, the mean radial coordinate and the second moment of $\theta_{\hat{1}}$. The numerical results plotted in this figure are given in Tables 4.1 and 4.2 in Appendix 4.B.

LogGaussian mixture (4.1) likelihoods. As for Figures 4.1 to 4.4 we use the `PolyChord` setting `num_repeats` = 20, which is deliberately chosen to be low in order to illustrate implementation-specific effects. The numerical results plotted in Figure 4.5 are given in Tables 4.1 and 4.2 in Appendix 4.B, along with the absolute values of the variation of results, root-mean-squared-errors and implementation error estimates. With these `PolyChord` settings, implementation-specific effects are the dominate source of parameter estimation errors for the LogGamma mixture likelihood. However, the implementation fraction of the error for the log-evidence calculations is significantly lower than for parameter estimation; this is because errors from the stochasticity of the nested sampling algorithm are much larger for evidence calculation than for parameter estimation.

The mean calculated value of $\log \mathcal{Z}$ for the LogGamma mixture likelihood (4.1), shown in Table 4.2, differs by 0.10 ± 0.03 from the true value from (4.3) of $\log \mathcal{Z}_{\text{true}} = -d \log(60)$. This systematic bias is due to `PolyChord` failing to consistently explore the posterior in this challenging case with the deliberately low setting `num_repeats` setting used — it can be reduced by increasing `num_repeats`. However despite the bias, our approach successfully detected implementation-specific effects in this case. Furthermore, using the true value, we can calculate an estimate of the implementation-specific effects

which accounts for the bias by replacing the standard deviation of results in (4.5) with the root-mean-squared-error (RMSE). Using the RMSE,

$$\sigma_{\text{imp,RMSE}} = \begin{cases} \sqrt{\text{RMSE}^2 - \sigma_{\text{bs}}^2} & \text{if RMSE}^2 > \sigma_{\text{bs}}^2, \\ 0 & \text{otherwise,} \end{cases} \tag{4.7}$$

where for some quantity with true value y_{true} the RMSE for a set of N calculated values $\hat{y}_1, \ldots, \hat{y}_N$ is

$$\text{RMSE} = \sqrt{\frac{\sum_{n=1}^{N} (\hat{y}_n - y_{\text{true}})^2}{N}}. \tag{4.8}$$

In this case the estimated $\sigma_{\text{imp}}/\sigma_{\text{values}}$ ratio of 0.43 ± 0.23 shown in Figure 4.5 is only a small underestimate compared to $\sigma_{\text{imp,RMSE}}/\text{RMSE} = 0.50 \pm 0.14$. Assessing results for systematic bias when an analytical value is not available is discussed in Section 4.6.3.

Skilling (2006) recommends that inferences from multiple nested sampling runs are made by combining them into a single run rather than simply averaging the results from each run, as this allows more accurate estimation of sample weights. If implementation-specific effects are negligible then uncertainty estimates can be calculated from the combined run using standard techniques, but this will be inaccurate if implementation-specific effects are the dominant source of error. In the latter case, the approximate error on the combined inference σ_{combined} from N nested sampling runs with the same settings can be roughly estimated as

$$\sigma_{\text{combined}} = \sigma_{\text{values}}/\sqrt{N}. \tag{4.9}$$

This may be an overestimate as it does not including the benefits of combining the runs, but in practice this effect is likely to be small compared to the uncertainty in the sample standard deviation of the separate runs σ_{values} unless N is very large.

4.5 Diagnostic tests for when few runs are available

For computationally expensive problems there may not be enough nested sampling runs available to calculate the implementation-specific effects directly using the method described in the previous section. In Sections 4.5.1 and 4.5.2 we therefore consider diagnostics which assess whether two nested sampling runs have consistently explored a parameter space while accounting for the stochastic nature of the nested sampling

algorithm. Due to the relatively small amount of information available in this case, it is useful to also consider qualitative comparisons using diagnostic plots of the types shown in Section 4.3 as well as any problem-specific knowledge of what the results should be. If $N > 2$ runs are available then $\binom{N}{2}$ pairwise tests can be computed and their results combined for greater accuracy.

4.5.1 Testing for correlations between threads in two different runs

We now introduce a test to assess whether nested sampling software is consistently exploring a posterior by comparing the statistical properties of the set of constituent threads (single live point runs) of two nested sampling runs. Each thread represents a valid nested sampling run and can be used to make posterior inferences about quantities such as the evidence and the mean and median of parameters. The actual values calculated from each thread will have large errors due their small number of samples, but this does not matter for testing if the distributions of values obtained from each run's threads are consistent.

We propose applying the 2-sample Kolmogorov-Smirnov (KS) test (Massey, 1951) to different runs' constituent threads by using each thread to calculate an estimate of a scalar quantity of interest (such as parameter means or the Bayesian evidence $\mathcal{Z}$) with the following procedure:

1. divide the first nested sampling run into its n_1 constituent threads, and calculate an estimate of the quantity from each;

2. divide the second nested sampling run into its n_2 constituent threads, and calculate an estimate of the quantity from each;

3. apply the 2-sample KS test to the n_1 and n_2 values calculated from the first and second runs respectively.

As a test statistic for distributions $p(x)$ and $q(x)$, the KS test uses the maximum distance between their cumulative distributions $F_p(x)$ and $F_q(x)$

$$D_{p,q} = \sup_x |F_p(x) - F_q(x)|, \tag{4.10}$$

where sup is the supremum. If n_1 and n_2 samples from $p(x)$ and $q(x)$ respectively are used, the corresponding p-values are

$$\alpha = 2 \exp\left(-\frac{2n_1 n_2}{n_1 + n_2} D_{p,q}^2\right). \tag{4.11}$$

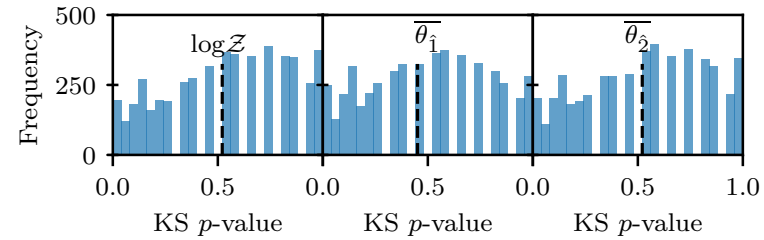
In this case the p-value produced represents the probability of observing a KS statistic $D_{p,q}$ of this size or greater if the threads in the two runs were drawn from the same distribution. A p-value close to zero implies that the values obtained from the threads in the two runs are statistically inconsistent, and hence that implementation-specific effects are likely to be present. This procedure can also be used with other distribution-free tests such as the 2-sample Anderson-Darling test (Scholz and Stephens, 1987) as an alternative to the KS test.

Figure 4.6 shows distributions of the p-values computed by applying this procedure to different pairs of nested sampling runs. For the LogGamma mixture likelihood the median p-values for $\overline{\theta_{\hat{1}}}$ and $\overline{\theta_{\hat{2}}}$ are 2×10^{-4} and 5×10^{-5} respectively, strongly suggesting that implementation-specific effects are present (in agreement with Figure 4.5). However, the approach is not able to detect significant evidence of implementation-specific effects in $\log \mathcal{Z}$ calculations, as implementation-specific effects comprise only a fraction of the total variation of results in this case so the pairs of runs do not provide enough information.

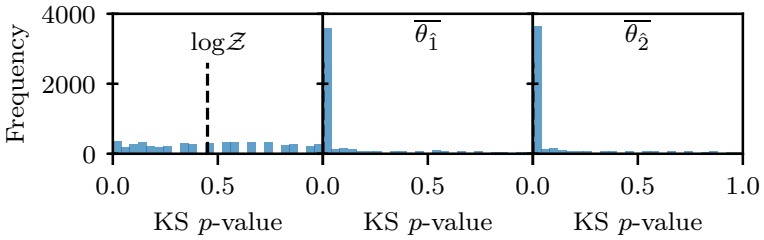
In addition there are many quantities which can be tested — for example the Bayesian evidence and the mean, median, higher moments and credible intervals of each parameter.[5] Considering a number of quantities allows sensitive testing for implementation-specific errors from only two nested sampling runs, even if the implementation-specific effects are smaller than in the LogGamma mixture case. One could also test multiple quantities together using a multi-dimensional KS test, although this is computationally challenging — see Fasano and Franceschini (1987) for a more detailed discussion.

For `MultiNest` runs using the setting `mmodal=True`, when a new mode is recognised, the run is split and live points assigned to the mode remain in that mode and evolve independently from the remainder of the run. As a result, even when there are no implementation-specific effects, the threads within such a run are not independently drawn from the same distribution and the KS test will not give correct p-values. The

[5] Tests on functions of the same parameter will not be independent.



(a) 10-dimensional Gaussian likelihood (3.7).



(b) 10-dimensional LogGamma mixture likelihood (4.1).

Figure 4.6: Distributions of KS p-values from pairwise comparison of different runs' constituent threads, using $\log \mathcal{Z}$ and the first two parameters. A p-value of 0 means the quantities calculated from threads in the two runs are from different distributions, implying the threads within each run are correlated with each other and implementation-specific effects are present. The black dashed line shows the median p-value for each plot. The nested sampling runs are the same ones that were used for Figure 4.5 — the 100 runs allow $\binom{100}{2} = 4,950$ pairwise statistics to be computed.

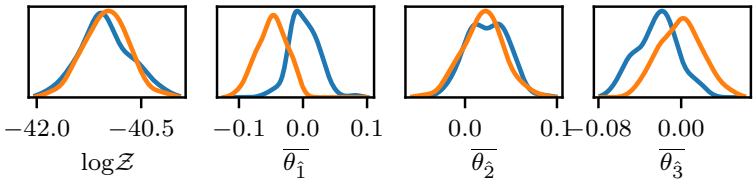
test is valid for `PolyChord` runs and `MultiNest` runs with `mmodal=False` as in these cases threads move between modes; this can be seen in Figure 4.4.

It is important to note that the KS p-value only determines whether implementation-specific effects are present and does not provide information about the size of implementation error, which must be assessed to determine if they are problematic for a given use case.[6] This can be done with the help of bootstrap resamples, as discussed in the next section.

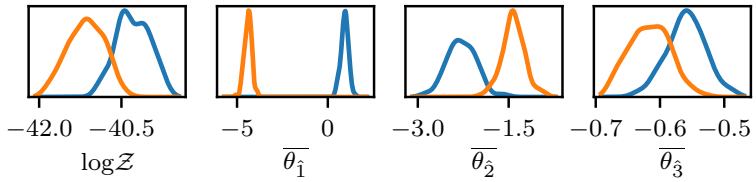
4.5.2 Distributions of sampling errors from bootstrap resamples

Our second diagnostic assesses whether calculations of scalar quantities from the two different runs differ by more than would be expected given the estimated uncertainties from the intrinsic stochasticity of the nested sampling algorithm. These uncertainty

[6] In particular with enough data (threads) one can get very low p-values even if the implementation-specific effects are relatively small and/or not important for the practical problem being examined.



(a) 10-dimensional Gaussian likelihood (3.7).



(b) 10-dimensional LogGamma mixture likelihood (4.1).

Figure 4.7: Plots of the sampling errors distribution calculated from bootstrap resampling threads for different quantities. Each plot shows 2 nested sampling runs (represented by different line colours), each with 250 live points and `num_repeats` = 20. The kernel density estimation of the posterior distributions use a Gaussian kernel with the bandwidth selected using Scott's rule (Scott, 2015).

distributions on posterior point estimates can be calculated from bootstrap resamples using the method described in Section 3.4.2, and are illustrated in Figures 4.7a and 4.7b. This has some similarities with Figures 4.2a and 4.2b but differs in that, in order to more easily quantify the comparison between runs, we are now considering only errors on single numbers rather than on whole posterior distributions. As a result this approach can also be applied to the Bayesian evidence $\mathcal{Z}$, which is a number rather than a distribution.

Bootstrap error distributions on point estimates for different runs like those in Figure 4.7 can be assessed for consistency qualitatively, or their tension can be quantified by calculating measures of the statistical distance between the distributions. As with the comparisons of threads in Section 4.5.1 it may be hard to draw conclusions from any one quantity, but the two runs can be compared using many different posterior estimates. Quantification may be more convenient than plotting graphs when comparing many different quantities or pairs of runs.

We use the KS statistic (4.10) as a statistical distance measure; this constitutes a metric as it is non-negative, zero if and only if the distributions are equal, symmetric and satisfies the triangle inequality. Its numerical values are also easy to interpret, with a value of 0 meaning the distributions are the same and a value of 1 meaning they

do not overlap. KS statistical distances between bootstrap uncertainty distributions on posterior point estimates are shown in Figure 4.8. These distributions show strong evidence for implementation-specific effects in parameter estimation for the LogGamma mixture case, with calculations of $\overline{\theta_{\hat{1}}}$ and $\overline{\theta_{\hat{2}}}$ having 65.7% and 67.9% of their pairwise statistical distances equalling 1 respectively. However, as for the diagnostic introduced in Section 4.5.1, two runs do not provide enough information to detect the relatively weaker implementation-specific effects in the LogGamma mixture $\log \mathcal{Z}$ estimates.

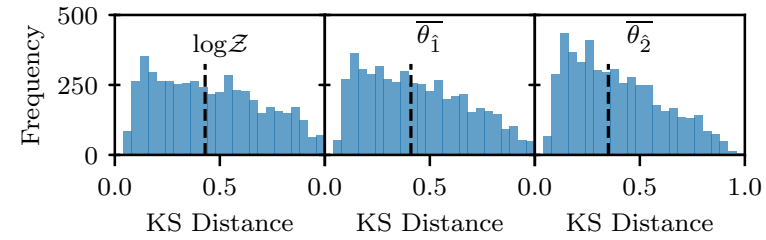
The KS statistical distances are more difficult to interpret than the p-values in Section 4.5.1, but have the advantage that together with plots like Figure 4.7 they contain information about the size of any implementation-specific effects as well as testing if implementation-specific effects are present. In this context, the KS statistic values are simply used as a distance measure and cannot be interpreted as p-values; even without implementation-specific effects, nested sampling runs will differ due to the stochasticity of the algorithm and so bootstrap resamples of different runs are always drawn from different distributions.

4.6 Implementation-specific effects in practice

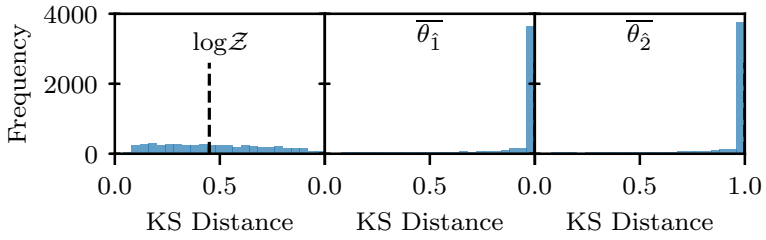
Having introduced our diagnostic tests, we now empirically test how different software settings and problem dimension affect the size of implementation-specific effects using `PolyChord`. The section finishes with a summary of our practical advice for software users.

4.6.1 Effect of sampling efficiency settings

Nested sampling software packages typically have settings controlling the process of sampling within a hard likelihood constraint which can reduce implementation-specific effects at the cost of increased computation. `PolyChord`'s `num_repeats` setting controls the number of slice samples taken before sampling each new live point — increasing this value reduces correlation between points and increases the accuracy with which `PolyChord` performs the nested sampling algorithm. Other examples of similar parameters include `MultiNest`'s `efr`, which controls the efficiency of its rejection sampling algorithm by determining the size of the ellipsoid within which `MultiNest` samples. If `efr` is lowered, samples are drawn from a larger ellipsoid, increasing the rejection rate whilst



(a) 10-dimensional Gaussian likelihood (3.7).

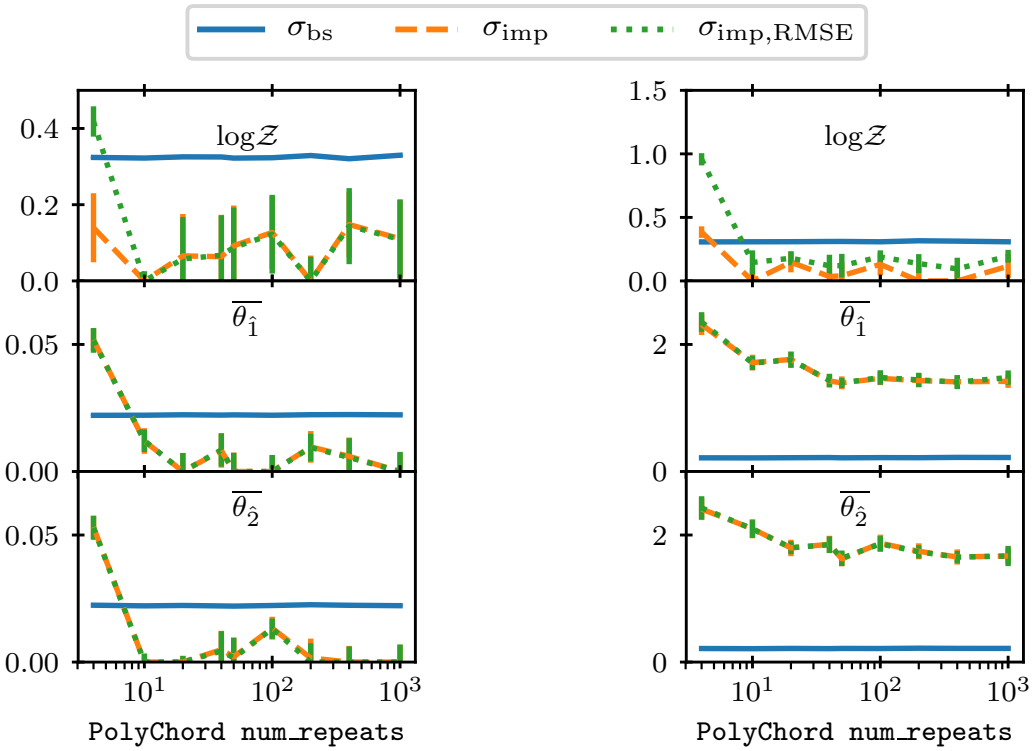


(b) 10-dimensional LogGamma mixture likelihood (4.1).

Figure 4.8: Distributions of KS statistical distances (4.10) between bootstrap uncertainty distributions on point estimates the type shown in Figure 4.7. For each likelihood, the 3 columns show results for $\log \mathcal{Z}$ calculations and for the mean of the parameters $\theta_{\hat{1}}$ and $\theta_{\hat{2}}$. The nested sampling runs are the same ones that were used for Figure 4.5; the 100 runs are compared pairwise to give $\binom{100}{2} = 4{,}950$ KS statistical distances for each quantity. A KS statistic of close to 1 means there is little overlap between the distributions, implying that the differences in the runs' values cannot be explained by the intrinsic stochasticity of the nested sampling algorithm and that implementation-specific effects are present. The black dashed line shows the median KS distance for each plot.

consequently decreasing the chance of missing part of the parameter space within the iso-likelihood contour. Hence, in contrast with `num_repeats`, implementation-specific effects are made smaller by *reducing* `efr`.

Figure 4.9 shows the effect on calculation errors of `PolyChord`'s `num_repeats` setting. As expected, we see that as `num_repeats` is increased the implementation-specific effects are reduced — showing `PolyChord` is performing the nested sampling algorithm with increasing accuracy. However, the `num_repeats` value required for implementation-specific effects to be a small fraction of the total error is highly problem dependent, even for the same number of dimensions. For the 10-dimensional Gaussian likelihood `num_repeats` $= 10$ is easily sufficient, but for the challenging 10-dimensional LogGamma likelihood `num_repeats` $> 10^3$ is needed. `num_repeats` can be tuned by, for example, doubling it until results show small implementation errors. In principle a sufficiently



(a) 10-dimensional Gaussian likelihood (3.7) with a uniform prior.

(b) 10-dimensional LogGamma mixture (4.1) with a uniform prior.

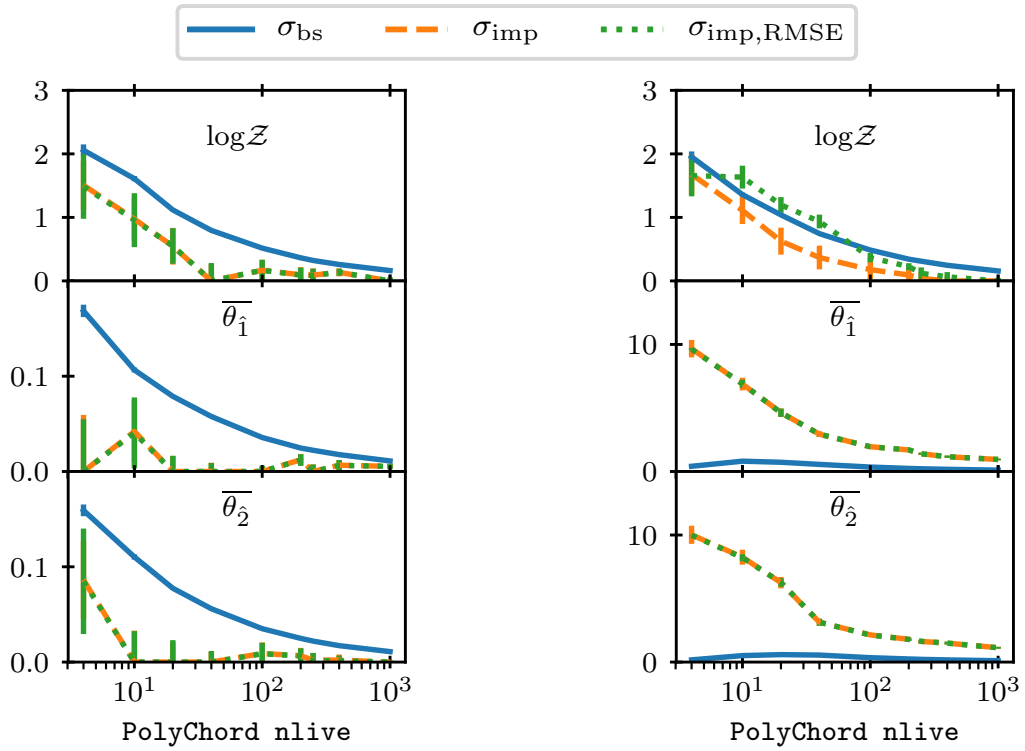
Figure 4.9: The effect of `PolyChord`'s `num_repeats` setting on results errors; each subfigure shows calculations of the log-evidence and the mean of the first two parameters. Results for every `num_repeats` value were calculated using 100 nested sampling runs, each with 250 live points. Blue solid lines show the mean bootstrap error estimate and orange dashed lines show implementation-specific effect estimates from (4.5). Green dotted lines show the implementation-specific effects calculated using the root-mean-squared-error (4.7); where the green and orange lines are equal, there is no systematic bias in the results. Error bars show the uncertainty on results for each `num_repeats` value considered.

high `num_repeats` value can make such errors negligible even for challenging likelihoods, but this will become impractically computationally expensive and gives diminishing returns in cases like the LogGamma mixture shown in Figure 4.9b. Once `num_repeats` is high enough that the calculations are not systematically biased, simply repeating the calculation many times is more efficient at improving accuracy.

4.6.2 Effect of the number of live points

In addition to software specific settings, the main choice a nested sampling user must make is the number of live points, which controls the resolution of sampling and is proportional to the expected number of samples produced. For simplicity we consider only runs with a constant number of live points n, although our conclusions also apply to dynamic nested sampling (introduced in Chapter 5) — in which the number of live points varies. The changes in calculation errors with changes in the number of live points used is shown in Figure 4.10. As expected, increasing the number of live points reduces the implementation-specific effects, as well as the errors from the stochasticity of the nested sampling algorithm (measured by bootstrap resampling) which are approximately proportional to $1/\sqrt{n}$. The fraction of the total error made up by implementation-specific effects does not necessarily decrease with increased n — this depends on how the implementation-specific effects scale with n. For the Gaussian likelihood, implementation-specific effects cause only a small part of the total variation of results, whereas for the more challenging LogGamma mixture likelihood they are the main source of errors.

Given that increasing n reduces both implementation-specific effects and errors from the stochasticity of the nested sampling algorithm, this is often a better way to reduce total errors for the same computational cost than increasing `num_repeats`. However while increasing n may make the absolute errors small enough for the given use case, it is not guaranteed to reduce the fraction of errors from implementation-specific effects; as a result techniques for estimating nested sampling errors which do not account for implementation-specific effects may not be accurate.



(a) 10-dimensional Gaussian likelihood (3.7) with a uniform prior.

(b) 10-dimensional LogGamma mixture (4.1) with a uniform prior.

Figure 4.10: The effect of the number of live points on errors in `PolyChord` calculations; the two subfigures both show calculations of the log-evidence and the mean of the first two parameters. Results for each number of live points considered were calculated using 100 nested sampling runs with `num_repeats` = 10. Blue solid lines show the mean bootstrap error estimate and orange dashed lines show implementation-specific effect estimates from (4.5). Green dotted lines show the implementation-specific effects calculated using the root-mean-squared-error (4.7); where the green and orange lines are equal, there is no systematic bias in the results. Error bars show 1σ uncertainties on results for each number of live points considered.

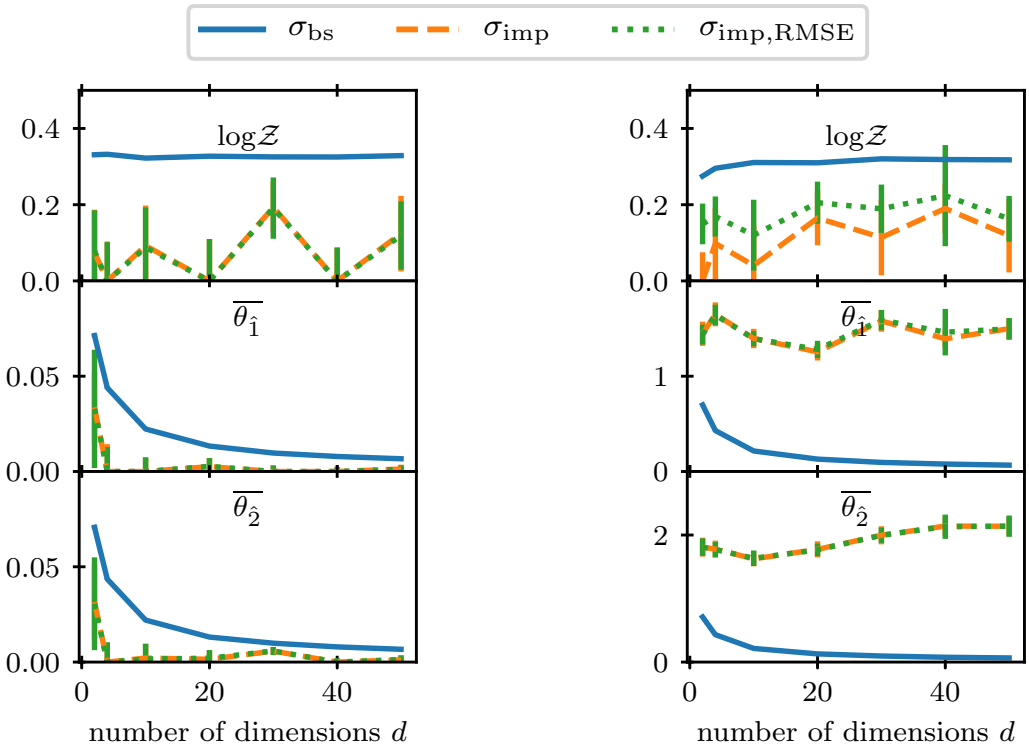
4.6.3 Calculation results with a systematic bias

Figures 4.9 and 4.10 show that for $\log \mathcal{Z}$ calculations, if `nlive` and `num_repeats` are set too low, estimates of the implementation-specific effects using the standard deviation of results and the root-mean-square error can start to differ. This is due to the algorithm failing to fully explore the posterior and iterating inwards too quickly, which leads to a systematic bias in $\log \mathcal{Z}$ (this is discussed in detail in Buchner, 2016). The `nlive` and `num_repeats` settings required to remove the bias depend on the posterior, with challenging multimodal or degenerate posteriors needing more samples (as for implementation-specific effects). The challenging LogGamma mixture likelihood shows a bias with the default `PolyChord` settings (as shown in Table 4.2 in Appendix 4.B), but this is small compared to the standard deviation of calculation results and can be reduced by increasing `num_repeats` or the number of live points. Systematic biases in a parameter estimation calculations are also possible with inappropriate settings, but in our experience this is much rarer.

The failure to fully explore the posterior which causes a systematic bias typically also results in differences between runs which are not explained by the stochasticity of the nested sampling algorithm — these implementation-specific effects can be detected by the diagnostic tests presented in this chapter. However, the bias causes these diagnostics to underestimate the size of the implementation-specific effects. If significant implementation-specific effects are detected in runs and the results of $\log \mathcal{Z}$ calculations are of interest, one can check for bias by repeating the calculation with higher `nlive` and `num_repeats` settings and checking if the mean calculated result changes.

4.6.4 Effect of dimensionality

Figure 4.11 shows implementation errors for the Gaussian and LogGamma mixture likelihoods for different numbers of dimensions d. Each calculation uses the `PolyChord` default settings of $25 \times d$ live points and `num_repeats` $= 5 \times d$; the defaults are proportional to d in order to give approximately constant errors in $\log Z$ (Handley et al., 2015b), with the additional samples produced for higher d leading to lower parameter estimation errors. With these settings, as d increases, our plot shows no strong upwards or downwards trend in the implementation error. Furthermore, the small bias in the $\log \mathcal{Z}$ calculation results for the LogGamma mixture likelihood (shown by the difference



(a) 10-dimensional Gaussian likelihood (3.7) with a uniform prior.

(b) 10-dimensional LogGamma mixture (4.1) with a uniform prior.

Figure 4.11: The effect of increasing the dimension d on errors in `PolyChord` calculations: each subfigure shows calculations of the log-evidence and the mean of the first two parameters. Results for every dimension d use the `PolyChord` default settings of $25 \times d$ live points and `num_repeats` $= 5 \times d$. Blue solid lines show the mean bootstrap error estimate and orange dashed lines show implementation-specific effect estimates from (4.5). Green dotted lines show the implementation-specific effects calculated using the root-mean-squared-error (4.7); where the green and orange lines are equal, there is no systematic bias in the results. Error bars show 1σ uncertainties on results for different numbers of dimensions.

between the green and orange lines in the top panel of Figure 4.11b) remains much smaller than the standard deviation of the results values $\sigma_{\text{values}} = \sqrt{\sigma_{\text{bs}}^2 + \sigma_{\text{imp}}^2}$.

4.6.5 Practical advice for software users

We finish by giving a summary of our approach to checking nested sampling calculations for challenging likelihoods where implementation errors may be present, based on our experience using nested sampling software.

We advise performing multiple nested sampling runs, and plotting the results to first assess their variation by eye as described in Section 4.3. If the results appear reasonable, one can perform a rough check for implementation-specific effects using the techniques described in Section 4.4 and/or Section 4.5, depending on how many runs are available. If implementation-specific errors are negligible:

- accuracy can be increased by simply calculating more runs and/or increasing the number of live points;
- the computational cost of future runs can be reduced by reducing the computational effort spent decorrelating samples (for example reducing `PolyChord`'s `num_repeats`, increasing `MultiNest`'s `efr` or changing the equivalent setting in the software package used). After large changes to the settings, the new results should be checked for implementation-specific effects;
- uncertainties on the results can be calculated using standard nested sampling methods such as the bootstrap resampling of threads, which will be accurate in this case.

In contrast, if implementation-specific effects are significant or are the dominant source of error:

- results should be recalculated with more live points and/or using more computational effort decorrelating samples (i.e. increasing `PolyChord`'s `num_repeats`, reducing `MultiNest`'s `efr` or changing the equivalent setting in the software used). If the calculation is already very computationally costly, increasing the number of live points is typically the best option as this will also reduce errors from the stochasticity of the nested sampling algorithm;

- there may be an additional systematic bias present in the results of evidence calculations. The mean calculated value for results using the new settings should be checked to see if it is significantly different to the mean result produced with the previous settings;
- the uncertainty on the combined results from the nested sampling runs can be roughly estimated from (4.9).

4.7 Application to Planck survey data

We now apply the tests introduced in this chapter to astronomical data from the *Planck* survey, which measures anisotropies in the cosmic microwave background (CMB). A detailed description of the associated cosmology and the ΛCDM concordance model is beyond the current scope; for this we refer the reader to Planck Collaboration (2013).

Given the ΛCDM concordance model, we can describe the universe's cosmology using only six parameters. Four of these are "late-time" parameters, governing the physics of the universe during and after reionisation: the present-day values of the Hubble constant H_0, the baryonic and cold dark matter fractions Ω_b and Ω_c, and the optical depth of the CMB τ. The remaining two parameters delineate the primordial universe through the amplitude A_s and tilt $n_\mathrm{s} - 1$ of the power spectrum of comoving curvature perturbations. To aid with MCMC sampling techniques, `cosmomc` (Lewis and Bridle, 2002) reparameterises the matter fractions as $\Omega_b h^2$ and $\Omega_c h^2$ in terms of the reduced Hubble constant h, defined by $H_0 = 100h$ km/s/Mpc, and in place of the Hubble constant uses $100\theta_{MC}$ (100× the ratio of the approximate sound horizon to the angular diameter distance). For more details about the parameters, see the first *Planck* parameters paper (Planck Collaboration, 2013).

Given a set of cosmological parameters, using a Boltzmann code such as `camb` (Lewis et al., 2000), one may compute theoretical CMB power spectra, which are then provided as inputs to cosmological likelihoods derived from CMB observations. We use the `Plik_lite` TT likelihood detailed by Planck Collaboration (2016b) and the default `CosmoChord` priors (see Handley et al., 2015a, for more information); these were used in Planck Collaboration (2016a). The likelihood introduces a single additional nuisance parameter for measurement calibration, increasing the dimensionality of the parameter space to seven.

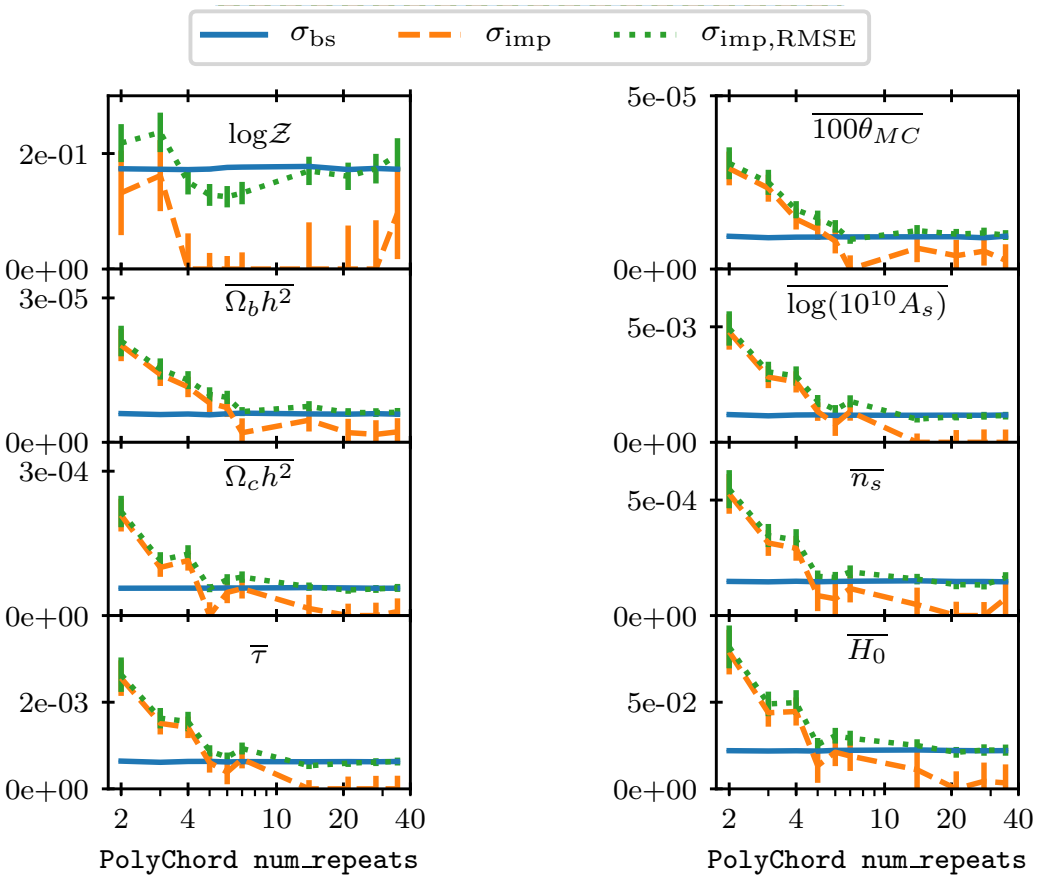


Figure 4.12: Implementation-specific effects in calculations using *Planck* data for different `PolyChord num_repeats` settings. The left column shows results for the evidence $\log \mathcal{Z}$ and the mean of the present day Baryon density $\Omega_b h^2$, present day cold matter density $\Omega_c h^2$ and Thompson scattering optical depth of the CMB τ. The right column shows results for calculations of the mean of the ratio of the sound horizon to angular distance (scaled by 100) $100\theta_{MC}$, the log power of the primordial curvature perturbations $\log(10^{10} A_s)$, the spectral index of the scalar primordial power spectrum n_s and the present day Hubble constant (derived from the other parameters) H_0. Results for every `num_repeats` value were calculated using 25 nested sampling runs, each with 500 live points. Blue solid lines show the mean bootstrap error estimate and orange dashed lines show implementation-specific effect estimates from (4.5). Green dotted lines show the implementation-specific effects calculated using the root-mean-squared-error (4.7); where the green and orange lines are equal, there is no systematic bias in the results. Error bars show the 1σ uncertainty on results for each `num_repeats` value considered.

Figure 4.12 shows estimates of implementation-specific effects for calculations using the *Planck* likelihoods and priors. As expected, there is a clear trend showing increasing `num_repeats` reduces implementation-specific effects. Furthermore in this case `PolyChord` default setting of `num_repeats` = 35 (5 times the number of dimensions) is sufficient to make such effects small or negligible for all the calculations shown.

However, as in the test cases in previous sections, significant implementation-specifics are present in the calculations if `num_repeats` is set too low. This is illustrated in Figure 4.13 for `num_repeats` = 1; with this setting the two runs (in red and blue) differ by more than the uncertainty expected from the stochasticity of the nested sampling algorithm shown by the coloured distributions. Such implementation-specific effects can also be detected in plots of the type introduced in Section 4.3.2 and with the diagnostic tests described in Section 4.5 (we do not show these for brevity).

It should be noted that in cosmology one traditionally uses likelihoods with many more nuisance parameters than in this analysis. One of the innovations that `PolyChord` provided to the *Planck* collaboration was its ability to exploit a fast-slow hierarchy of parameter speeds (Lewis, 2013). In this context, nuisance parameters that do not require recomputation of expensive parts of the likelihood may be varied at negligible cost in comparison with the slower cosmological parameters. Increasing the number of steps in nuisance parameters directions greatly aids mixing and the reduction of implementation-specific errors. However, a full analysis of this specific case is beyond the scope of this chapter.

4.8 Conclusion

In this chapter we introduced diagnostic tests for nested sampling software, which uses numerical techniques to generate approximately uncorrelated samples within hard likelihood constraints. As a result, for challenging problems such as those with multimodal or degenerate posteriors, additional errors may be produced which would not be present if the nested sampling algorithm was performed perfectly; we term these implementation-specific effects. Detecting the presence of significant implementation-specific effects is of great importance for software users as it determines whether results and estimates of uncertainties can be relied upon, and if the settings should be changed.

We suggested two new diagnostic diagrams for visualising nested sampling results

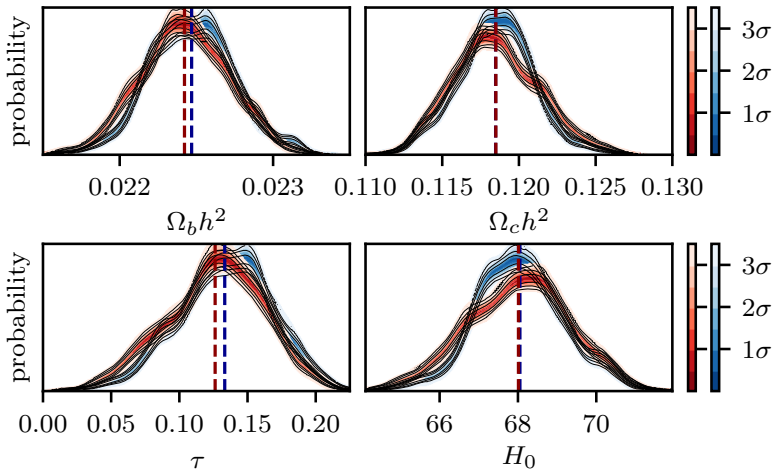


Figure 4.13: As for Figure 4.2 but using the *Planck* survey likelihood and prior. The first row shows the present day Baryon density $\Omega_b h^2$ and the present day cold matter density $\Omega_c h^2$; the second row shows the optical depth of the CMB τ and the present day Hubble constant H_0. Each run uses 500 live points, and has `num_repeats` $= 1$ — the low value is chosen to illustrate implementation-specific effects. The coloured contours show iso-probability credible intervals on the marginalised posterior probability density function at each parameter value due to the stochasticity of the nested sampling algorithm. The dashed dark blue and dark red lines show the estimated posterior means of each parameter for the blue and red runs respectively.

and uncertainties and for comparing nested sampling runs; these are shown in Figures 4.2 to 4.4. Section 4.4 introduced a quantitative measure of implementation-specific effects, which can be used to estimate them directly if enough nested sampling runs are available to estimate the standard deviation of results. In addition, Section 4.5 provided two diagnostic tests which can be applied with only two nested sampling runs. We find that due to the larger errors from the stochasticity of the nested sampling algorithm in evidence calculations, implementation-specific errors form a smaller fraction of the total error in this case — and are consequently less important and harder to detect than in parameter estimation.

In Section 4.6 we empirically tested the effects of software settings and the number of dimensions on implementation-specific effects, and discussed dealing with cases where nested sampling results are systematically biased. Our practical advice for nested sampling software users based on our experience is summarised in Section 4.6.5. Finally, Section 4.7 demonstrated the application of our diagnostics to an astronomical problem using data from the *Planck* survey.

	$\log \mathcal{Z}$	$\overline{\theta_{\hat{1}}}$	$\overline{\theta_{\hat{2}}}$	$\overline{\theta_{\hat{3}}}$
Analytic Value	-40.9434	0.0000	0.0000	0.0000
Mean Result	-40.93(3)	0.002(2)	0.000(2)	0.000(2)
σ_{values}	0.33(2)	0.022(2)	0.019(1)	0.019(1)
σ_{bs}	0.326(3)	0.0223(2)	0.0223(2)	0.0221(2)
σ_{imp}	0.07(11)	0.000(7)	0.000(3)	0.000(3)
$\sigma_{\text{imp}}/\sigma_{\text{values}}$	0.20(33)	0.00(34)	0.00(17)	0.00(17)
Values RMSE	0.33(2)	0.022(2)	0.019(1)	0.019(1)
$\sigma_{\text{imp,RMSE}}$	0.06(11)	0.000(7)	0.000(2)	0.000(3)
$\sigma_{\text{imp,RMSE}}$/RMSE	0.17(33)	0.00(34)	0.00(17)	0.00(19)

Table 4.1: Calculation error results for the 100 nested sampling runs with a Gaussian likelihood shown in Figure 4.5. The first two rows shows the analytical value for each estimator and the mean calculation result. The next three rows show the bootstrap error estimate, implementation error estimate (4.5) and the ratio of the implementation estimate to the standard deviation of results. The final three rows show the root-mean-squared-error, the implementation-specific effects estimate from (4.7), and the ratio of the two. Columns show results for the log-evidence and the mean of the first three parameters. Numbers in parentheses show the 1σ numerical uncertainty on the final digit.

We have written a publicly available software package `nestcheck` (Higson, 2018b), which performs diagnostics on input nested sampling runs and produces plots like Figures 4.2 to 4.4; it can be downloaded at `https://github.com/ejhigson/nestcheck`.

Appendix 4.A Code

The code used to perform the numerical tests and generate the results in this chapter can be downloaded at `https://github.com/ejhigson/diagnostic`; this provides examples of the use of the `nestcheck` package.

Appendix 4.B Numerical results tables

Tables 4.1 and 4.2 given numerical results for the nested sampling runs plotted in Figure 4.5.

	$\log \mathcal{Z}$	$\overline{\theta_{\hat{1}}}$	$\overline{\theta_{\hat{2}}}$	$\overline{\theta_{\hat{3}}}$
Analytic Value	-40.9434	-0.5772	0.0000	-0.5772
Mean Result	-40.84(3)	-0.49(18)	-0.22(18)	-0.572(3)
σ_{values}	0.34(2)	1.78(13)	1.81(13)	0.032(2)
Values RMSE	0.36(2)	1.77(12)	1.81(10)	0.032(2)
σ_{bs}	0.309(3)	0.217(2)	0.215(2)	0.0300(3)
σ_{imp}	0.15(8)	1.76(13)	1.80(13)	0.01(1)
$\sigma_{\text{imp}}/\sigma_{\text{values}}$	0.43(23)	0.993(1)	0.993(1)	0.31(30)
$\sigma_{\text{imp,RMSE}}$	0.18(6)	1.76(13)	1.80(10)	0.011(9)
$\sigma_{\text{imp,RMSE}}$/RMSE	0.50(14)	0.992(1)	0.9930(8)	0.33(28)

Table 4.2: As in Table 4.1 but for calculations using the LogGamma mix likelihood (4.1).

Chapter 5

Dynamic nested sampling

This chapter introduces dynamic nested sampling: a generalisation of the nested sampling algorithm in which the number of live points varies to allocate samples more efficiently. In empirical tests the new method significantly improves calculation accuracy compared to standard nested sampling with the same number of samples; this increase in accuracy is equivalent to speeding up the computation by factors of up to ~ 72 for parameter estimation and ~ 7 for evidence calculations. Several dynamic nested sampling software packages are now publicly available[1] and the algorithm has been applied to a variety of astronomical problems. This chapter is an edited version of Higson et al. (2019a).

5.1 Introduction

Nested sampling explores the posterior distribution by maintaining a set of samples from the prior, called *live points*, and iteratively updating them subject to the constraint that new samples have increasing likelihoods. Conventionally a fixed number of live points is used; we term this *standard nested sampling*. In this case the expected fractional shrinkage of the prior volume remaining is the same at each step, and as a result many samples are typically taken from regions of the prior that are remote from the bulk

[1] Dynamic nested sampling packages include:
`dyPolyChord` (`https://github.com/ejhigson/dyPolyChord`); Python, C++ and Fortran likelihoods and priors, based on `PolyChord`.
`dynesty` (`https://github.com/joshspeagle/dynesty`); pure Python.
`perfectns` (`https://github.com/ejhigson/perfectns`); pure Python, spherically symmetric likelihoods and priors only.

of the posterior. The allocation of samples in standard nested sampling is set by the likelihood and the prior, and cannot be changed depending on whether calculating the evidence or obtaining posterior samples is the primary goal.

We propose modifying the nested sampling algorithm by dynamically varying the number of live points in order to maximise the accuracy of a calculation for some number of posterior samples, subject to practical constraints. We term this more general approach *dynamic nested sampling*, with standard nested sampling representing the special case where the number of live points is constant. Dynamic nested sampling is particularly effective for parameter estimation, as standard nested sampling typically spends most of its computational effort iterating towards the posterior peak. This produces posterior samples with negligible weights which make little contribution to parameter estimation calculations, as discussed in Chapter 3. We also achieve significant improvements in the accuracy of evidence calculations, and show both evidence and parameter estimation can be improved simultaneously. Our approach can be easily incorporated into existing standard nesting sampling software; we have created the `dyPolyChord` package (Higson, 2018a) for performing dynamic nested sampling using `PolyChord`.

In this chapter we demonstrate the advantages of dynamic nested sampling relative to the popular standard nested sampling algorithm in a range of empirical tests. An empirical comparison of nested sampling with alternative methods such as MCMC-based parameter estimation and thermodynamic integration is beyond the current scope — for this we refer the reader to Allison and Dunkley (2014), Murray (2007) and Feroz (2008).

The chapter begins with an overview of some related methods, then Section 5.2 establishes useful results about the effects of varying the number of live points. Our dynamic nested sampling algorithm for increasing efficiency in general nested sampling calculations is presented in Section 5.3; its accurate allocation of live points for *a priori* unknown posterior distributions is illustrated in Figure 5.3. We first test dynamic nested sampling using `perfectns` (Higson, 2018c), which able to perform perfect nested sampling in both standard and dynamic versions. This allows us to conduct a wide range of tests without prohibitive computational costs, and avoids implementation-specific effects — which will vary between different software packages. These tests are described in Section 5.4, which includes a discussion of the effects of likelihood, priors and dimensionality on the improvements from dynamic nested sampling. In particular

we find large efficiency gains for high-dimensional parameter estimation problems.

Section 5.5 discusses applying dynamic nested sampling to challenging posteriors, in which results from nested sampling software may include implementation-specific effects (see Chapter 4 for a detailed discussion). We describe the strengths and weaknesses of dynamic nested sampling compared to standard nested sampling in such cases, and perform numerical tests with a multimodal Gaussian mixture model using `dyPolyChord`. We find that dynamic nested sampling also produces significant accuracy gains for this more challenging posterior, and that it is able to reduce implementation-specific effects compared to standard nested sampling.

5.1.1 Other related work

Other variants of nested sampling include diffusive nested sampling (Brewer et al., 2011) and superposition enhanced nested sampling (Martiniani et al., 2014), which were mentioned in Section 2.4.2 and have been implemented as stand-alone software packages. In particular, dynamic nested sampling shares some similarities with `DNest4` (Brewer and Foreman-Mackey, 2018), in which diffusive nested sampling is followed by additional sampling targeting regions of high posterior mass. However dynamic nested sampling differs from these alternatives as, like standard nested sampling, it only requires drawing samples within hard likelihood constraints. As a result dynamic nested sampling can be used to improve the efficiency of popular standard nested sampling implementations such as `MultiNest` (rejection sampling), `PolyChord` (slice sampling) and constrained Hamiltonian nested sampling (Betancourt, 2011) while maintaining their strengths in sampling degenerate and multimodal distributions.

It has been shown that efficiency can be greatly increased using nested importance sampling (Chopin and Robert, 2010) or by performing nested sampling using an auxiliary prior which approximates the posterior as described in Cameron and Pettitt (2014). However, the efficacy of these approaches is contingent on having adequate knowledge of the posterior (either before the algorithm is run, or by using the results of previous runs). As such, the speed increase on *a priori* unknown problems is generally lower than might be suggested by toy examples.

Dynamic nested sampling is similar in spirit to the adaptive schemes for thermodynamic integration introduced by Hug et al. (2016) and Friel et al. (2014), as each involves an initial run followed by additional targeted sampling using an estimated er-

ror criteria. Furthermore, dynamically weighting sampling in order to target regions of higher posterior mass has also been used in the statistical physics literature, such as in multi-canonical sampling (see for example Okamoto, 2004).

5.2 Variable numbers of live points

Before presenting our dynamic nested sampling algorithm in Section 5.3, we first establish some basic results for a nested sampling run in which the number of live points varies. Such runs are valid as successive shrinkage ratios t_i are independently distributed (Skilling, 2006). For now we assume the manner in which the number of live points changes is specified in advance; adaptive allocation of samples is considered in the next section.

Let us define n_i as the number of live points present for the prior shrinkage ratio t_i between dead points $i-1$ and i.[2] In this notation all information about the number of live points for a nested sampling run can be expressed as a list of numbers $\mathbf{n} = \{n_1, n_2, \ldots, n_{n_{\mathrm{dead}}}\}$ which correspond to the shrinkage ratios $\mathbf{t} = \{t_1, t_2, \ldots, t_{n_{\mathrm{dead}}}\}$. n_i and $\mathbf{n}$ are distinguished from the symbol n used to denote a constant number of live points in previous chapters by their subscript and bold font respectively. Nested sampling calculations for variable numbers of live points differ from the constant live point case only in the use of different n_i in calculating the distribution of each t_i from (2.18).

Skilling (2006)'s method for combining constant live point runs, mentioned in Section 3.4.1, can be extended to accommodate variable numbers of live points by requiring that at any likelihood value the live points of the combined run equals the sum of the live points of the constituent runs at that likelihood value (this is illustrated in Figure 5.1). Variable live point runs can also be divided into their constituent threads using the algorithm in Section 3.4.1. However, unlike for constant live point runs, the threads produced may start and finish part way through the run and there is no longer a single unique division into threads on iso-likelihood contours where the number of live points increases. The technique for estimating sampling errors by resampling threads introduced in Section 3.4.2 can also be applied for nested sampling runs with variable

[2] In order for (2.18) to be valid, the number of live points must remain constant across the shrinkage ratios t_i between successive dead points. We therefore only allow the number of live points to change on iso-likelihood contours $\mathcal{L}(\boldsymbol{\theta}) = \mathcal{L}_i$ where a dead point i is present. This restriction has negligible effects for typical calculations, and is automatically satisfied by most nested sampling implementations.

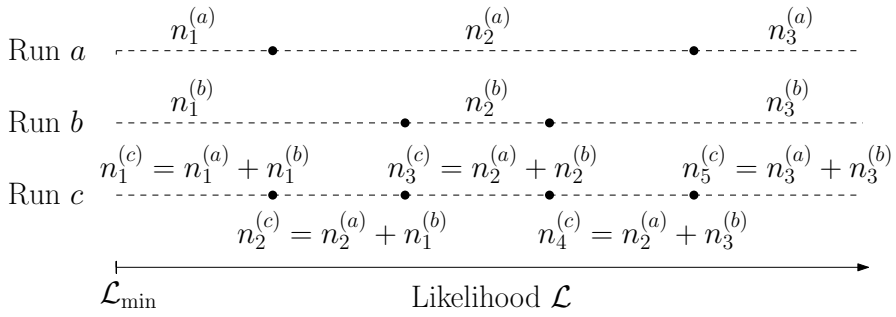


Figure 5.1: Combining nested sampling runs a and b with variable numbers of live points $\mathbf{n}^{(a)}$ and $\mathbf{n}^{(b)}$ into a single nested sampling run c; black dots show dead points arranged in order of increasing likelihood. The number of live points in run c at some likelihood value equals the sum of the live points of run a and run b at that likelihood value.

numbers of live points (see Appendix 5.B for more details), as can the diagnostic tests for implementation-specific effects described in Chapter 4.

In addition, the variable live point framework provides a natural way to include the final set of live points remaining when a standard nested sampling run terminates in a calculation. These are uniformly distributed in the region of the prior with $\mathcal{L}(\boldsymbol{\theta}) > \mathcal{L}_{\text{terminate}}$, and can be treated as samples from a dynamic nested sampling run with the number of live points reducing by 1 as each of the points remaining after termination is passed until the final point i has $n_i = 1$. This allows the final live points of standard nested sampling runs to be combined with variable live point runs.

The remainder of this section analyses the effects of local variations in the number of live points on the accuracy of nested sampling evidence calculation and parameter estimation. The dynamic nested sampling algorithm in Section 5.3 uses these results to allocate additional live points.

5.2.1 Effects on calculation accuracy

Nested sampling calculates the evidence $\mathcal{Z}$ as the sum of sample weights (2.19); the dominant sampling errors are from statistically estimating shrinkage ratios t_i which affect the weights of all subsequent points. In Appendix 5.C we show analytically that the reduction in evidence errors achieved by taking additional samples to increase the local number of live points n_i is inversely proportional to n_i, and is approximately proportional to the evidence contained in point i and all subsequent points. This makes

sense as the dominant evidence errors are from statistically estimating shrinkages t_i which affect all points $j \geq i$.

In nested sampling parameter estimation, sampling errors come both from taking a finite number of samples in any region of the prior and from the stochastic estimation of their normalised weights p_i from (2.20). Typically standard nested sampling takes many samples with negligible posterior mass as illustrated in Figure 2.1; these make little contribution to estimates of parameters or to the accuracy of samples' normalised weights. From (2.18) the expected separation between points in $\log X$ (approximately proportional to the posterior mass they each represent) is $1/n_i$. As a result, increasing the number of live points wherever the dead points' posterior weights $p_i \propto \mathcal{L}_i w_i$ are greatest distributes posterior mass more evenly among the samples. This improves the accuracy of the statistically estimated weights p_i, and can dramatically increase the information content (Shannon entropy of the samples)

$$H = \exp\left(-\sum_i p_i \log p_i\right), \tag{5.1}$$

which is maximised for a given number of samples when the sample weights are equal. Empirical tests of dynamic nested sampling show that increasing the number of live points wherever points have the highest $p_i \propto \mathcal{L}_i w_i$ works well as regards increasing parameter estimation accuracy for most calculations.

As the contribution of each sample i to a parameter estimation problem for some quantity $f(\boldsymbol{\theta})$ is dependent on $f(\boldsymbol{\theta}_i)$, the precise optimum allocation of live points is different for different quantities. In most cases the relative weight p_i of samples is a good approximation for their influence on a calculation, but for some problems much of the error may come from sampling $\log X$ regions containing a small fraction of the posterior mass but with extreme parameter values. Appendix 5.D discusses estimating the importance of points to a specific parameter estimation calculation and using dynamic nested sampling to allocate live points accordingly.

5.3 The dynamic nested sampling algorithm

This section presents our algorithm for performing nested sampling calculations with a dynamically varying number of live points to optimise the allocation of samples.

Since the distribution of posterior mass as a function of the likelihood is *a priori* unknown, we first approximate it by performing a standard nested sampling run with some small constant number of live points n_{init}. The algorithm then proceeds by iteratively calculating the range of likelihoods where increasing the number of live points will have the greatest effect on calculation accuracy, and generating an additional thread running over these likelihoods. If required some n_{batch} additional threads can be generated at each step to reduce the number of times the importance must be calculated and the sampler restarted. We find in empirical tests that using $n_{\text{batch}} > 1$ has little effect on efficiency gains from dynamic nested sampling when the number of samples taken in each batch is small compared to the total number of samples in the run.

From the discussion in Section 5.2.1 we define functions to measure the relative importance of a sample i for evidence calculation and parameter estimation respectively as

$$I_{\mathcal{Z}}(i) \propto \frac{\mathrm{E}[\mathcal{Z}_{\geq i}]}{n_i}, \quad \text{where } \mathcal{Z}_{\geq i} \equiv \sum_{k \geq i} \mathcal{L}_k w_k(\mathbf{t}), \tag{5.2}$$

$$I_{\text{param}}(i) \propto \mathcal{L}_i \ \mathrm{E}[w_i(\mathbf{t})]. \tag{5.3}$$

Alternatively (5.2) can be replaced with the more complex expression (5.22) derived in Appendix 5.C, although we find this typically makes little difference to results. Modifying (5.3) to optimise for estimation of a specific parameter or function of parameters is discussed in Appendix 5.D.

The user specifies how to divide computational resources between evidence calculation and parameter estimation through an input goal $G \in [0, 1]$, where $G = 0$ corresponds to optimising for evidence calculation and $G = 1$ optimises for parameter estimation. The dynamic nested sampling algorithm calculates importance as a weighted sum of the points' normalised evidence and parameter estimation importances

$$I(G, i) = (1 - G)\frac{I_{\mathcal{Z}}(i)}{\sum_j I_{\mathcal{Z}}(j)} + G\frac{I_{\text{param}}(i)}{\sum_j I_{\text{param}}(j)}. \tag{5.4}$$

The likelihood range in which to run an additional thread is chosen by finding all points with importance greater than some fraction f of the largest importance. Choosing a smaller fraction makes the threads added longer and reduces the number of times the importance must be recalculated, but can also cause the number of live points to plateau for regions with importance greater than that fraction of the maximum importance (see

the discussion of Figure 5.3 in the next section for more details). We use $f = 0.9$ for results in this chapter, but find empirically that using slightly higher or lower values make little difference to results. To ensure any steep or discontinuous increases in the likelihood $\mathcal{L}(X)$ are captured we find the first point j and last point k which meet this condition, then generate an additional thread starting at $\mathcal{L}_{j-1}$ and ending when a point is sampled with likelihood greater than $\mathcal{L}_{k+1}$. If j is the first dead point, threads which initially sample the whole prior are generated. If k is the final dead point then the thread will stop when a sample with likelihood greater than $\mathcal{L}_k$ is found.[3] This allows the new thread to continue beyond $\mathcal{L}_k$, meaning dynamic nested sampling iteratively explores higher likelihoods when this is the most effective use of samples.

Unlike in standard nested sampling, more accurate dynamic nested sampling results can be obtained simply by continuing the calculation for longer. The user must specify a condition at which to stop dynamically adding threads, such as when fixed number of samples has been taken or some desired level of accuracy has been achieved. Sampling errors on evidence and parameter estimation calculations can be estimated from the dead points at any stage using the method described in Chapter 3. We term these *dynamic termination conditions* to distinguish them from the type of termination conditions used in standard nested sampling. Our dynamic nested sampling algorithm is presented more formally in Algorithm 4.

Output : Samples and live points information **n**.
Input : Goal G, n_{init}, dynamic termination condition.

Generate a nested sampling run with a constant number of live points n_{init};
while *dynamic termination condition not satisfied* **do**
  recalculate importance $I(G, i)$ of all points;
  find first point j and last point k with importance of greater than some fraction f (we use $f = 0.9$) of the largest importance;
  generate an additional thread (or alternatively n_{batch} additional threads) starting at $\mathcal{L}_{j-1}$ and ending with the first sample taken with likelihood greater than $\mathcal{L}_{k+1}$[4];
end

Algorithm 4: Dynamic nested sampling.

[3] We find empirically that one additional point per thread is sufficient to reach higher likelihoods if required. This is because typically there are many threads, and for each thread (which has only one live point) the expected shrinkage between samples (2.18) of $E[\log t_i] = -1$ is quite large.

5.3.1 Software implementation

Since dynamic nested sampling only requires the ability to sample from the prior within a hard likelihood constraint, implementations and software packages developed for standard nested sampling can be easily adapted to perform dynamic nested sampling. We demonstrate this with the `dyPolyChord` package, which performs dynamic nested sampling using `PolyChord` and is compatible with Python, C++ and Fortran likelihoods.

`PolyChord` was designed before the creation of the dynamic nested sampling algorithm, and is not optimized to quickly resume the nested sampling process at an arbitrary point to add more threads. `dyPolyChord`, which performs nested sampling with `PolyChord`, minimises the computational overhead from saving and resuming by using Algorithm 5 — a modified version of Algorithm 4 described in Appendix 5.F. After the initial exploratory run with n_{init} live points, Algorithm 5 calculates a dynamic allocation of live points and then generates more samples in a single run without recalculating point importances. This means only the initial run provides information on where to place samples, and as a result the allocation of live points is slightly less accurate and a higher value of n_{init} is typically needed.

Dynamic nested sampling will be incorporated in the forthcoming `PolyChord 2` software package, which is currently in development and is designed for problems of up to $\sim 1,000$ dimensions — dynamic nested sampling can provide very large improvements in the accuracy of such high-dimensional problems, as shown by the numerical tests in the next section. Furthermore, we anticipate reloading a past iteration i of a `PolyChord 2` nested sampling run in order to add additional threads will be less computationally expensive than a single likelihood call for many problems. Nevertheless, it is often more efficient for dynamic nested sampling software to generate additional threads in selected likelihood regions in batches rather than one at a time; this approach is used in the `dynesty`[5] dynamic nested sampling package.

[4] If k is the final dead point, the additional thread terminates after the first point with likelihood greater than $\mathcal{L}_k$.

[5] See `https://github.com/joshspeagle/dynesty` for more information.

5.4 Numerical tests with perfect nested sampling

As discussed in Chapter 3, perfect nested sampling calculations depend on the likelihood $\mathcal{L}(\boldsymbol{\theta})$ and prior $\pi(\boldsymbol{\theta})$ only through the distribution of posterior mass $\mathcal{L}(X)$ and the distribution of parameters on iso-likelihood contours $P(f(\boldsymbol{\theta})|\mathcal{L}(\boldsymbol{\theta}) = \mathcal{L}(X))$, each of which is a function of both $\mathcal{L}(\boldsymbol{\theta})$ and $\pi(\boldsymbol{\theta})$. We therefore empirically test dynamic nested sampling using likelihoods and priors with a wide range of distributions of posterior mass, and consider a variety of functions of parameters $f(\boldsymbol{\theta})$ in each case.

We first examine perfect nested sampling of d-dimensional spherical unit Gaussian likelihoods centred on the origin

$$\mathcal{L}(\boldsymbol{\theta}) = (2\pi)^{-d/2}\mathrm{e}^{-|\boldsymbol{\theta}|^2/2}. \qquad \text{(3.7 revisited)}$$

For additional tests using distributions with lighter and heavier tails we use d-dimensional exponential power likelihoods

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{d\,\Gamma(\frac{d}{2})}{\pi^{\frac{d}{2}}2^{1+\frac{1}{2b}}\Gamma(1+\frac{n}{2b})}\mathrm{e}^{-|\boldsymbol{\theta}|^{2b}/2}, \qquad (5.5)$$

where $b = 1$ corresponds to a d-dimensional Gaussian (3.7). All tests use d-dimensional co-centred spherical Gaussian priors

$$\pi(\boldsymbol{\theta}) = (2\pi\sigma_\pi^2)^{-d/2}\mathrm{e}^{-|\boldsymbol{\theta}|^2/2\sigma_\pi^2}. \qquad \text{(3.9 revisited)}$$

The different distributions of posterior mass in $\log X$ for (3.7) and (5.5) with dimensions d are illustrated in Figure 5.2. As in previous chapters we denote the first component of the $\boldsymbol{\theta}$ vector as $\theta_{\hat{1}}$, although by symmetry the results will be the same for any component. $\overline{\theta_{\hat{1}}}$ is the mean of the posterior distribution of $\theta_{\hat{1}}$, and the one-tailed $Y\%$ upper credible interval $\text{C.I.}_{Y\%}(\theta_{\hat{1}})$ is the value $\theta^*_{\hat{1}}$ for which $P(\theta_{\hat{1}} < \theta^*_{\hat{1}}|\mathcal{L},\pi) = Y/100$.

Tests of dynamic nested sampling terminate after a fixed number of samples, which is set such that they use similar or slightly smaller numbers of samples than the standard nested sampling runs we compare them to. Dynamic runs have n_{init} set to 10% of the number of live points used for the standard runs. Standard nested sampling runs use the termination conditions described by Handley et al. (2015b, Section 3.4), stopping when the estimated evidence contained in the live points is less than 10^{-3} times the evidence contained in dead points (the default value used in `PolyChord`). This is an appropriate termination condition for nested sampling parameter estimation, but if only the evidence

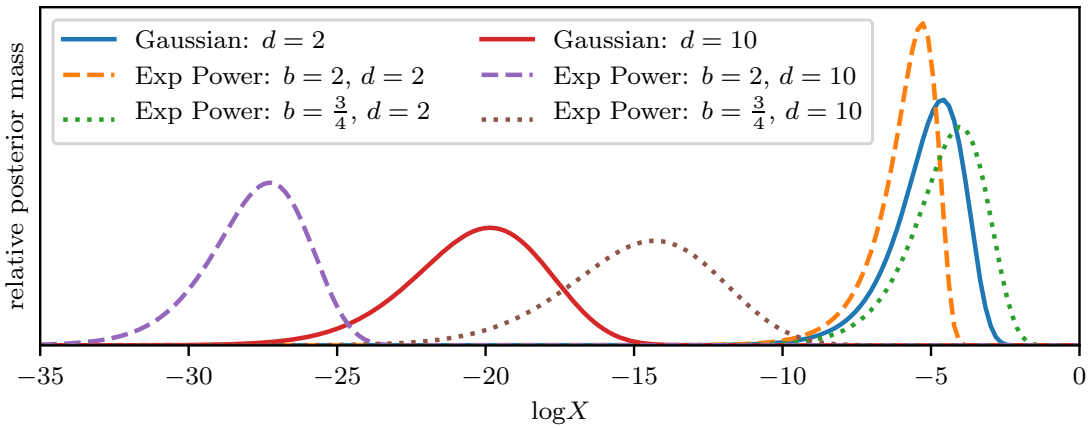


Figure 5.2: Relative posterior mass ($\propto \mathcal{L}(X)X$) as a function of $\log X$ for Gaussian likelihoods (3.7) and exponential power likelihoods (5.5) with $b = 2$ and $b = \frac{3}{4}$. Each has a Gaussian prior (3.9) with $\sigma_\pi = 10$. The lines are scaled so that the area under each of them is equal.

is of interest then stopping with a larger fraction of the posterior mass remaining will have little effect on calculation accuracy.

The increase in computational efficiency from our method can be calculated by observing that nested sampling calculation errors are typically proportional to the square root of the computational effort applied (Skilling, 2006), and that the number of samples produced is approximately proportional to the computational effort. The increase in efficiency (computational speedup) from dynamic nested sampling over standard nested sampling for runs containing approximately the same number of samples on average can therefore be estimated from the variation of results as

$$\text{efficiency gain} = \frac{\text{Var}\left[\text{standard NS results}\right]}{\text{Var}\left[\text{dynamic NS results}\right]}. \tag{5.6}$$

Here the numerator is the variance of the calculated values of some quantity (such as the evidence or the mean of a parameter) from a number of standard nested nested sampling runs, and the denominator is the variance of the calculated values of the same quantity from a number of dynamic nested sampling runs. When the two methods use different numbers of samples on average, (5.6) can be replaced with

$$\text{efficiency gain} = \frac{\text{Var}\left[\text{standard NS results}\right]}{\text{Var}\left[\text{dynamic NS results}\right]} \times \frac{\overline{N_{\text{samp,sta}}}}{\overline{N_{\text{samp,dyn}}}}, \tag{5.7}$$

where the additional term is the ratio of the mean number of samples produced by the standard and dynamic nested sampling runs.

5.4.1 10-dimensional Gaussian example

We begin by testing dynamic nested sampling on a 10-dimensional Gaussian likelihood (3.7) with a Gaussian prior (3.9) and $\sigma_\pi = 10$. Figure 5.3 shows the relative allocation of live points as a function of $\log X$ for standard and dynamic nested sampling runs. The dynamic nested sampling algorithm (Algorithm 4) can accurately and consistently allocate live points, as can be seen by comparison with the analytically calculated distribution of posterior mass and posterior mass remaining. Dynamic nested sampling live point allocations do not precisely match the distribution of posterior mass and posterior mass remaining in the $G = 1$ and $G = 0$ cases because they include the initial exploratory run with a constant n_{init} live points. Furthermore as additional live points are added where the importance is more than 90% of the maximum importance, the number of live points allocated by dynamic nested sampling is approximately constant for regions with importance of greater than $\sim 90\%$ of the maximum — this can be clearly seen in Figure 5.3 near the peak number of live points in the $G = 1$ case. Similar diagrams for exponential power likelihoods (5.5) with $b = 2$ and $b = \frac{3}{4}$ are provided in Appendix 5.E.1 (Figures 5.12 and 5.13), and show the allocation of live points is also accurate in these cases.

The variation of results from repeated standard and dynamic nested sampling calculations with a similar number of samples is shown in Table 5.1 and Figure 5.4. Dynamic nested sampling optimised for evidence calculation ($G = 0$) and parameter estimation ($G = 1$) produce significantly more accurate results than standard nested sampling. In addition, results for dynamic nested sampling with $G = 0.25$ show that both evidence calculation and parameter estimation accuracy can be improved simultaneously. Equivalent results for 10-dimensional exponential power likelihoods (5.5) with $b = 2$ and $b = \frac{3}{4}$ are shown in Tables 5.6 and 5.7 in Appendix 5.E.1. The reduction in evidence errors for $G = 0$ and parameter estimation errors for $G = 1$ in Table 5.1 correspond to increasing efficiency by factors of 1.40 ± 0.04 and up to 4.4 ± 0.1 respectively.

5.4.2 Efficiency gains for different distributions of posterior mass

Efficiency gains (5.6) from dynamic nested sampling depend on the fraction of the $\log X$ range explored which contains samples that make a significant contribution to calculation accuracy. If this fraction is small most samples taken by standard nested sampling

	samples	$\log \mathcal{Z}$	$\overline{\theta_{\hat{1}}}$	median($\theta_{\hat{1}}$)
St.Dev. standard	15,189	0.189(2)	0.0158(2)	0.0194(2)
St.Dev. $G = 0$	15,152	0.160(2)	0.0180(2)	0.0249(2)
St.Dev. $G = 0.25$	15,156	0.179(2)	0.0124(1)	0.0163(2)
St.Dev. $G = 1$	15,161	0.549(5)	0.00834(8)	0.0104(1)
Gain $G = 0$		1.40(4)	0.77(2)	0.60(2)
Gain $G = 0.25$		1.11(3)	1.62(5)	1.42(4)
Gain $G = 1$		0.119(3)	3.6(1)	3.5(1)
(continued)	C.I.$_{84\%}(\theta_{\hat{1}})$	$\overline{\lvert\boldsymbol{\theta}\rvert}$	median($\lvert\boldsymbol{\theta}\rvert$)	
St.Dev. standard	0.0253(3)	0.0262(3)	0.0318(3)	
St.Dev. $G = 0$	0.0301(3)	0.0292(3)	0.0335(3)	
St.Dev. $G = 0.25$	0.0204(2)	0.0205(2)	0.0239(2)	
St.Dev. $G = 1$	0.0132(1)	0.0138(1)	0.0152(2)	
Gain $G = 0$	0.71(2)	0.80(2)	0.90(3)	
Gain $G = 0.25$	1.54(4)	1.64(5)	1.77(5)	
Gain $G = 1$	3.7(1)	3.6(1)	4.4(1)	

Table 5.1: Test of dynamic nested sampling for a 10-dimensional Gaussian likelihood (3.7) and a Gaussian prior (3.9) with $\sigma_\pi = 10$. The first row shows the standard deviation of $5,000$ calculations for standard nested sampling with a constant number of live points $n = 500$. The next three rows show the standard deviations of $5,000$ dynamic nested sampling calculations with a similar number of samples; these are respectively optimised purely for evidence calculation accuracy ($G = 0$), for both evidence and parameter estimation ($G = 0.25$) and purely for parameter estimation ($G = 1$). The final three rows show the computational efficiency gain (5.6) from dynamic nested sampling over standard nested sampling in each case. The first column shows the mean number of samples for the $5,000$ runs. The remaining columns show calculations of the log evidence, the mean, median and 84% one-tailed credible interval of a parameter $\theta_{\hat{1}}$, and the mean and median of the radial coordinate $|\boldsymbol{\theta}|$. Numbers in brackets show the 1σ numerical uncertainty on the final digit.

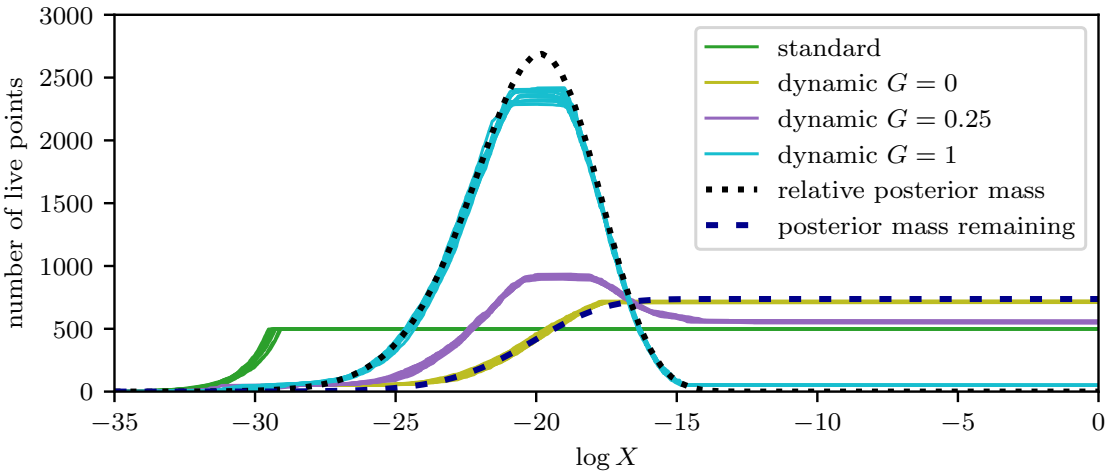


Figure 5.3: Live point allocation for a 10-dimensional Gaussian likelihood (3.7) with a Gaussian prior (3.9) and $\sigma_\pi = 10$. Solid lines show the number of live points as a function of $\log X$ for 10 standard nested sampling runs with $n = 500$, and 10 dynamic nested sampling runs with $n_{\text{init}} = 50$, a similar number of samples and different values of G. The dotted and dashed lines show the relative posterior mass $\propto \mathcal{L}(X)X$ and the posterior mass remaining $\propto \int_{-\infty}^{X} \mathcal{L}(X')X' \,\mathrm{d}X'$ at each point in $\log X$; for comparison these lines are scaled to have the same area under them as the average of the number of live point lines. Standard nested sampling runs include the final set of live points at termination, which are modeled using a decreasing number of live points as discussed in Section 5.2. Similar diagrams for exponential power likelihoods (5.5) with $b = 2$ and $b = \frac{3}{4}$ are presented in Figures 5.12 and 5.13 in Appendix 5.E.1.

contain little information, and dynamic nested sampling can greatly improve performance. For parameter estimation ($G = 1$), only $\log X$ regions containing significant posterior mass ($\propto \mathcal{L}(X)X$) are important, whereas for evidence calculation ($G = 0$) all samples taken before the bulk of the posterior is reached are valuable. Both cases benefit from dynamic nested sampling using fewer samples to explore the region after most of the posterior mass has been passed but before termination.

We now test the efficiency gains (5.6) of dynamic nested sampling empirically for a wide range of distributions of posterior mass by considering Gaussian likelihoods (3.7) and exponential power likelihoods (5.5) of different dimensions d and prior sizes σ_π. The results are presented in Figures 5.5 and 5.6, and show large efficiency gains from dynamic nested sampling for parameter estimation in all of these cases.

Increasing the dimension d typically means the posterior mass is contained in a smaller fraction of the prior volume, as shown in Figure 5.2. In the spherically symmetric

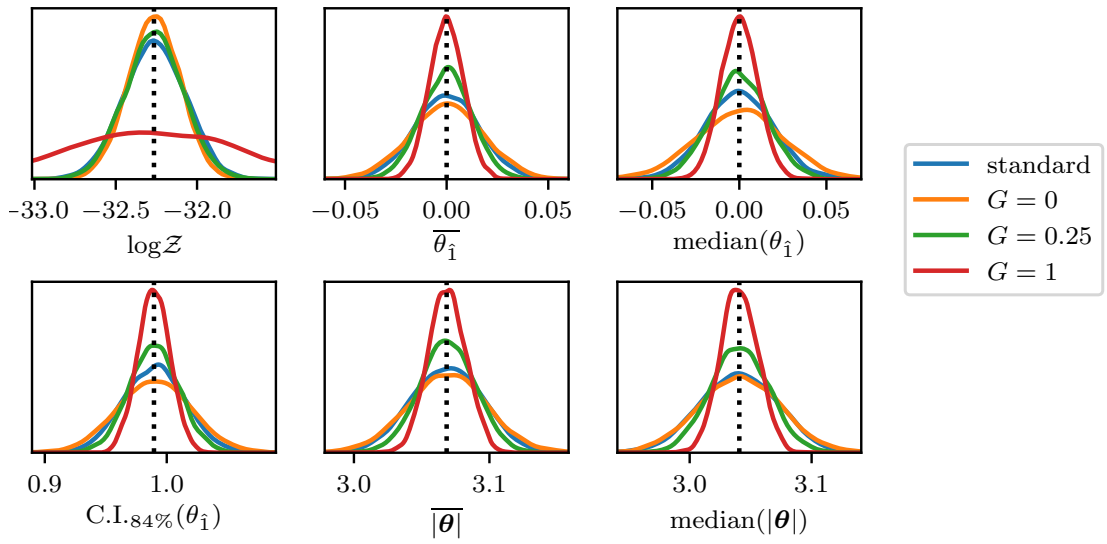


Figure 5.4: Distributions of results for the dynamic and standard nested sampling calculations shown in Table 5.1, plotted using kernel density estimation. Black dotted lines show the correct value of each quantity for the likelihood and prior used. Compared to standard nested sampling (blue lines), the distributions of results of dynamic nested sampling with $G = 1$ (red lines) for parameter estimation problems show much less variation around the correct value. Results for dynamic nested sampling with $G = 0$ (orange lines) are on average closer to the correct value than standard nested sampling for calculating $\log \mathcal{Z}$, and results with $G = 0.25$ (green lines) show improvements over standard nested sampling for both evidence and parameter estimation calculations.

cases we consider, the range of $\log X$ to be explored before significant posterior mass is reached increases approximately linearly with d. This increases the efficiency gain (5.6) from dynamic nested sampling for parameter estimation ($G = 1$) but reduces it for evidence calculation ($G = 0$). In high-dimensional problems the vast majority of the $\log X$ range explored is usually covered before any significant posterior mass is reached, resulting in very large efficiency gains for parameter estimation but almost no gains for evidence calculation — as can be seen in Figure 5.5. For the 1,000-dimensional exponential power likelihood with $b = 2$, dynamic nested sampling with $G = 1$ improves parameter estimation efficiency by a factor of up to 72 ± 5, with the largest improvement for estimates of the median the posterior distribution of $|\boldsymbol{\theta}|$.

Increasing the size of the prior σ_π increases the fraction of the $\log X$ range explored before any significant posterior mass is reached, resulting in larger efficiency gains (5.6) from dynamic nested sampling for parameter estimation ($G = 1$) but smaller gains for

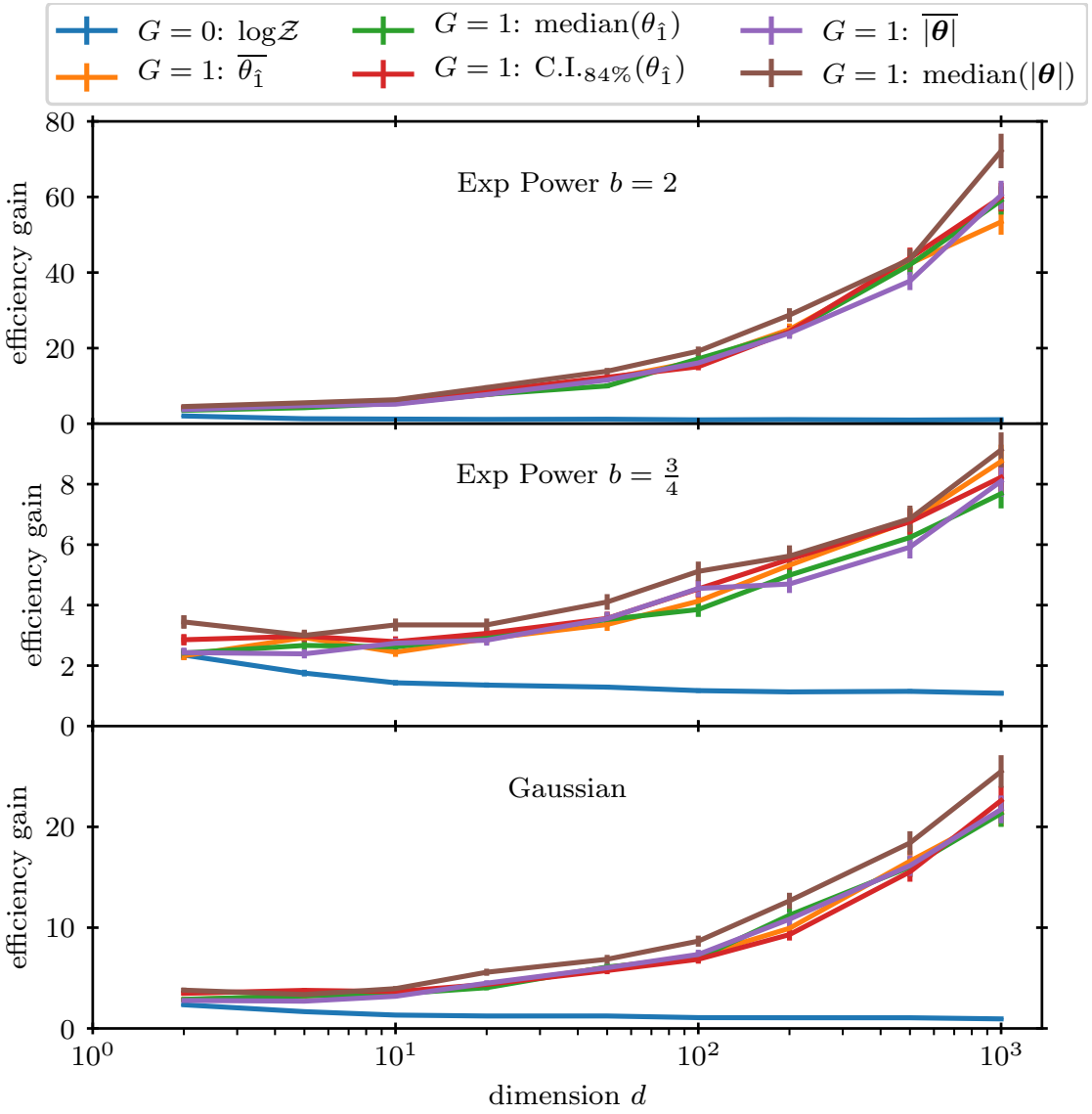


Figure 5.5: Efficiency gain (5.6) from dynamic nested sampling compared to standard nested sampling for likelihoods of different dimensions; each has a Gaussian prior (3.9) with $\sigma_\pi = 10$. Results are shown for calculations of the log evidence, the mean, median and 84% one-tailed credible interval of a parameter $\theta_{\hat{1}}$, and the mean and median of the radial coordinate $|\boldsymbol{\theta}|$. Each efficiency gain is calculated using 1,000 standard nested sampling calculations with $n = 200$ and 1,000 dynamic nested sampling calculations with $n_{\text{init}} = 20$ using a similar or slightly smaller number of samples.

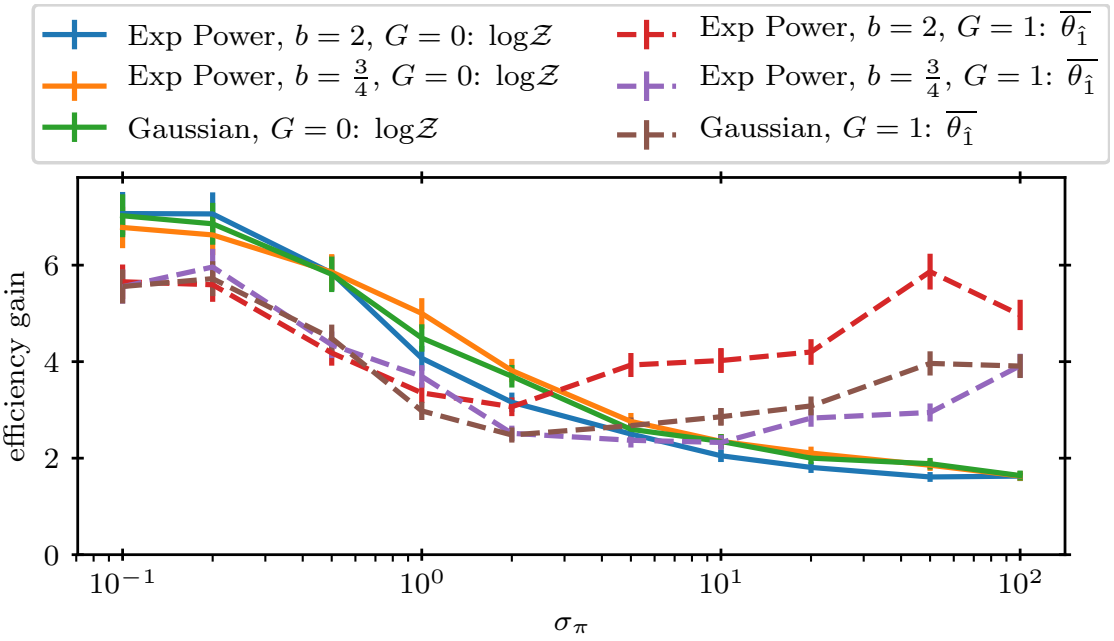


Figure 5.6: Efficiency gain (5.6) from dynamic nested sampling for Gaussian priors (3.9) of different sizes σ_π. Results are shown for calculations of the log evidence and the mean of a parameter $\theta_{\hat{1}}$ for 2-dimensional Gaussian likelihoods (3.7) and 2-dimensional exponential power likelihoods (5.5) with $b = 2$ and $b = \frac{3}{4}$. Each efficiency gain is calculated using 1,000 standard nested sampling calculations with $n = 200$ and 1,000 dynamic nested sampling calculations with $n_{\text{init}} = 20$ using a similar or slightly smaller number of samples.

evidence calculation ($G = 0$). However when σ_π is small the bulk of the posterior mass is reached after a small number of steps, and most of the $\log X$ range explored is after the majority of the posterior mass but before termination. Dynamic nested sampling places fewer samples in this region than standard nested sampling, leading to large efficiency gains for both parameter estimation and evidence calculation. This is shown in Figure 5.6; when $\sigma_\pi = 0.1$, dynamic nested sampling evidence calculations with $G = 0$ improve efficiency over standard nested sampling by a factor of approximately 7 for all 3 likelihoods considered. However we note that if only the evidence estimate is of interest then standard nested sampling can safely terminate with a higher fraction of the posterior mass remaining than 10^{-3}, in which case efficiency gains would be lower.

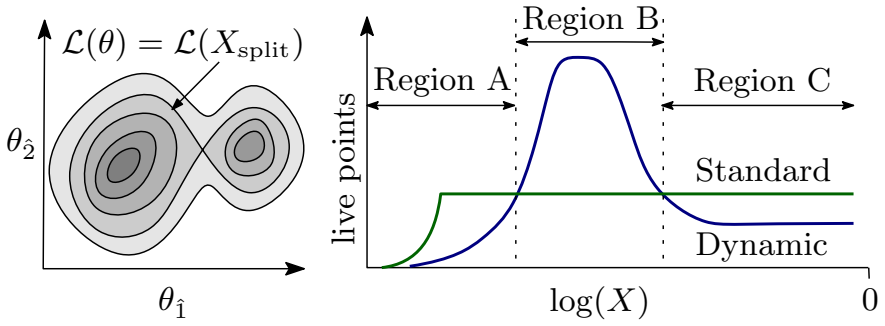


Figure 5.7: Dynamic and standard nested sampling's relative ability to discover hard to locate modes is determined by the number of live points present at the likelihood $\mathcal{L}(X_{\text{split}})$ at which a mode splits from the remainder of the posterior (illustrated on the left). In the schematic graph on the right we would expect dynamic nested sampling to be better at finding modes than standard nested sampling in region B (where it has a higher number of live points) but worse in regions A and C.

5.5 Dynamic nested sampling with challenging posteriors

Nested sampling software generally uses the population of dead and live points to sample within iso-likelihood contours, and so taking more samples in the region of an iso-likelihood contour will reduce the sampler's implementation-specific effects. As a result dynamic nested sampling typically has smaller implementation-specific effects than standard nested sampling in the regions of the posterior where it has a higher number of live points, but conversely may perform worse in regions with fewer live points. For highly multimodal or degenerate likelihoods it is important all modes or other regions of significant posterior mass are found by the sampler — dynamic nested sampling performs better than standard nested sampling at finding hard to locate modes which become separated from the remainder of the posterior at likelihood values where it has more live points,[6] as illustrated schematically in Figure 5.7.

Provided no significant modes are lost we expect dynamic nested sampling to have lower implementation-specific effects than standard nested sampling, as it has more live points — and therefore lower implementation-specific effects — in the regions which have the largest effect on calculation accuracy. If modes separate at likelihood values where dynamic nested sampling assigns few samples, n_{init} must be made large enough to ensure no significant modes are lost. For highly multimodal posteriors, a safe approach

[6]However, if a mode is only discovered late in the dynamic nested sampling process then it may still be under-sampled due to not being present in threads calculated before it was found.

is to set n_{init} high enough to find all significant modes, in which case dynamic nested sampling will use the remaining computational budget to minimise calculation errors. Even if, for example, half of the computational budget is used on the initial exploratory run, dynamic nested sampling will still achieve over half of the efficiency gain compared to standard nested sampling that it could with a very small n_{init}.

5.5.1 Numerical tests with a multimodal posterior

We now use `dyPolyChord` to numerically test dynamic nested sampling on a challenging multimodal d-dimensional, M-component Gaussian mixture likelihood

$$\mathcal{L}(\boldsymbol{\theta}) = \sum_{m=1}^{M} W^{(m)} \left(2\pi {\sigma^{(m)}}^2\right)^{-d/2} \exp\left(-\frac{|\boldsymbol{\theta} - \boldsymbol{\mu}^{(m)}|^2}{2{\sigma^{(m)}}^2}\right). \tag{5.8}$$

Here each component m is centred on a mean $\boldsymbol{\mu}^{(m)}$ with standard deviation $\sigma^{(m)}$ in all dimensions, and the component weights $W^{(m)}$ satisfy $\sum_{m=1}^{M} W^{(m)} = 1$. For comparison with the perfect nested sampling results using a Gaussian likelihood (3.7) in Section 5.4, we use $d = 10$, $\sigma^{(m)} = 1$ for all m and a Gaussian prior (3.9) with $\sigma_\pi = 10$. We consider a Gaussian mixture (5.8) of $M = 4$ components with means and weights

$$\begin{aligned}
W^{(1)} &= 0.4, & \mu^{(1)}_{\hat{1}} &= 0, & \mu^{(1)}_{\hat{2}} &= 4,\\
W^{(2)} &= 0.3, & \mu^{(2)}_{\hat{1}} &= 0, & \mu^{(2)}_{\hat{2}} &= -4,\\
W^{(3)} &= 0.2, & \mu^{(3)}_{\hat{1}} &= 4, & \mu^{(3)}_{\hat{2}} &= 0,\\
W^{(4)} &= 0.1, & \mu^{(4)}_{\hat{1}} &= -4, & \mu^{(4)}_{\hat{2}} &= 0,
\end{aligned} \tag{5.9}$$

$$\text{and } \mu^{(m)}_{\hat{k}} = 0 \quad \text{for all } k \in (3, \ldots, d),\ m \in (1, \ldots, M).$$

The posterior distribution for this case is shown in Figure 5.8.

As in Section 5.4, we compare standard nested sampling runs to dynamic nested sampling runs which use a similar or slightly smaller number of samples. `dyPolyChord` uses Algorithm 5, meaning only the initial run provides information on where to place samples, so we set n_{init} to 20% of the number of live points used in standard nested sampling runs they are compared to, instead of the 10% used in the perfect nested sampling tests in Section 5.4.

The allocation of live points from `dyPolyChord` runs with the Gaussian mixture likelihood (5.8) is shown in Figure 5.9. As in the tests with perfect nested sampling,

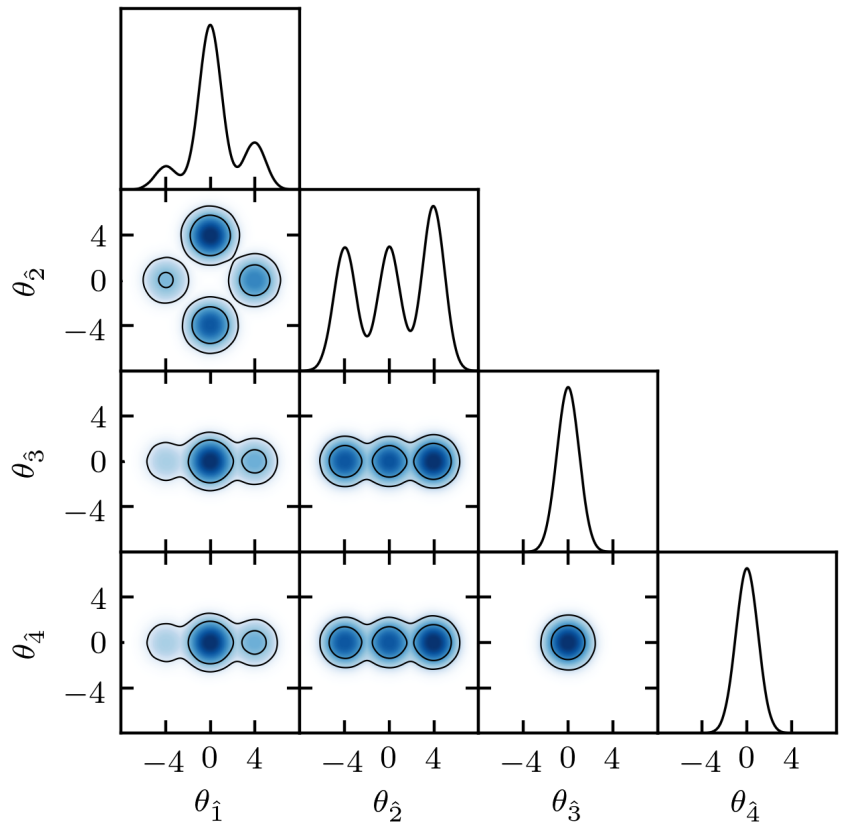


Figure 5.8: Posterior distributions for the 4-component 10-dimensional Gaussian mixture model (5.8) with component weights and means given by (5.9), and a Gaussian prior (3.9). By symmetry the distributions of $\theta_{\hat{k}}$ are the same for $k \in (3, \ldots, d)$, so we only show only the first 4 components of $\boldsymbol{\theta}$; 1- and 2-dimensional plots of other parameters are the same as those of $\theta_{\hat{3}}$ and $\theta_{\hat{4}}$.

the numbers of live points with settings $G = 1$ and $G = 0$ match the posterior mass and posterior mass remaining respectively despite the more challenging likelihood. The live point allocation is not as precise as in Figure 5.3 due to `dyPolyChord` only using information from the initial exploratory run to calculate all the point importances. Another difference is that the truncation of the peak number of live points in the $G = 1$ in Figure 5.3 is not present for `dyPolyChord` runs, as this is due to Algorithm 4 adding new points where the importance is within 90% of the maximum.

Table 5.2 shows the variation of repeated calculations for dynamic nested sampling for the 10-dimensional Gaussian mixture model (5.8) with `dyPolyChord`. This shows

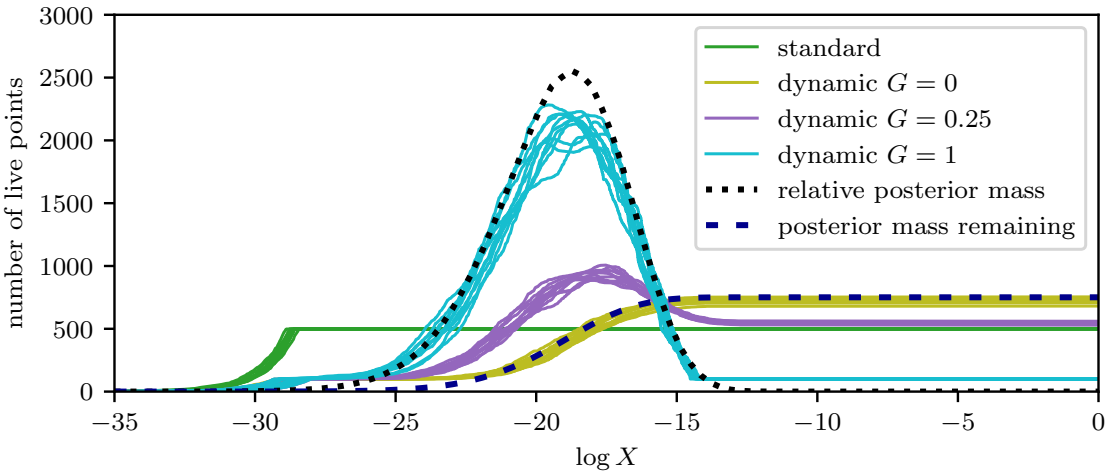


Figure 5.9: Live point allocation as in Figure 5.3 but with a 10-dimensional Gaussian mixture likelihood (5.8), with component weights and means given by (5.9) and a Gaussian prior (3.9) with $\sigma_\pi = 10$. The 10 standard nested sampling runs shown were generated using `PolyChord` with $n = 500$, and 10 dynamic nested sampling runs with each G value were generated using `dyPolyChord` with a similar number of samples and $n_{\text{init}} = 100$. The dotted and dashed lines show the relative posterior mass $\propto \mathcal{L}(X)X$ and the posterior mass remaining $\propto \int_{-\infty}^{X} \mathcal{L}(X')X' \,\mathrm{d}X'$ at each point in $\log X$; for comparison these lines are scaled to have the same area under them as the average of the number of live point lines.

significant efficiency gains (5.6) from dynamic nested sampling of 1.3 ± 0.1 for evidence calculation with $G = 0$ and up to 4.0 ± 0.4 for parameter estimation with $G = 1$, demonstrating how dynamic nested sampling can be readily applied to more challenging multimodal cases. In Appendix 5.E.2 we empirically verify that dynamic nested sampling does not introduce any errors from sampling bias (which would not be captured by efficiency gains (5.6) based on the variation of results) using analytically calculated true values of the log evidence and posterior means. Table 5.8 shows that the mean calculation results are very close to the correct values, and hence the standard deviation of the results is almost identical to their root-mean-squared-error, meaning efficiency gains (5.6) accurately reflect reductions in calculation errors (as for perfect nested sampling).

Table 5.3 shows estimated implementation-specific effects for the results in Table 5.2; these are calculated using the procedure described in Chapter 4, which estimates the part of the variation of results which is not explained by the intrinsic stochasticity of

	samples	$\log \mathcal{Z}$	$\overline{\theta_{\hat{1}}}$	$\overline{\theta_{\hat{2}}}$
St.Dev. standard	14,739	0.181(6)	0.057(2)	0.126(4)
St.Dev. $G = 0$	14,574	0.160(5)	0.076(2)	0.176(6)
St.Dev. $G = 0.25$	14,628	0.170(5)	0.046(1)	0.105(3)
St.Dev. $G = 1$	14,669	0.36(1)	0.032(1)	0.069(2)
Gain $G = 0$		1.3(1)	0.56(5)	0.51(5)
Gain $G = 0.25$		1.1(1)	1.5(1)	1.5(1)
Gain $G = 1$		0.25(2)	3.3(3)	3.4(3)
(continued)	median($\theta_{\hat{1}}$)	C.I.$_{84\%}$($\theta_{\hat{1}}$)	$\overline{\lvert\boldsymbol{\theta}\rvert}$	
St.Dev. standard	0.035(1)	0.170(5)	0.0196(6)	
St.Dev. $G = 0$	0.048(2)	0.229(7)	0.0222(7)	
St.Dev. $G = 0.25$	0.0293(9)	0.138(4)	0.0156(5)	
St.Dev. $G = 1$	0.0203(6)	0.085(3)	0.0110(3)	
Gain $G = 0$	0.53(5)	0.55(5)	0.78(7)	
Gain $G = 0.25$	1.4(1)	1.5(1)	1.6(1)	
Gain $G = 1$	3.0(3)	4.0(4)	3.2(3)	

Table 5.2: Tests of dynamic nested sampling as in Table 5.1 but with a 10-dimensional Gaussian mixture likelihood (5.8), with component weights and means given by (5.9) and a Gaussian prior (3.9) with $\sigma_\pi = 10$. The first row shows the standard deviation of 500 `PolyChord` standard nested sampling calculations with a constant number of live points $n = 500$. The next three rows show the standard deviations of 500 `dyPolyChord` calculations with a similar number of samples; these are respectively optimised purely for evidence calculations ($G = 0$), for both evidence and parameter estimation ($G = 0.25$) and purely for parameter estimation ($G = 1$). The final three rows show the computational efficiency gain (5.6) from dynamic nested sampling over standard nested sampling in each case. The first column shows the mean number of samples produced by the 500 runs. The remaining columns show calculations of the log evidence, the mean of parameters $\theta_{\hat{1}}$ and $\theta_{\hat{2}}$, the median and 84% one-tailed credible interval of $\theta_{\hat{1}}$, and the mean radial coordinate $|\boldsymbol{\theta}|$. Numbers in brackets show the 1σ numerical uncertainty on the final digit.

	$\log \mathcal{Z}$	$\overline{\theta_{\hat{1}}}$	$\overline{\theta_{\hat{2}}}$
Implementation St.Dev. standard	0.02(4)	0.044(2)	0.115(4)
Implementation St.Dev. $G = 0$	0.06(2)	0.062(3)	0.163(6)
Implementation St.Dev. $G = 0.25$	0.03(4)	0.035(2)	0.095(4)
Implementation St.Dev. $G = 1$	0.00(8)	0.024(1)	0.062(2)
(continued)	median($\theta_{\hat{1}}$)	C.I.$_{84\%}$($\theta_{\hat{1}}$)	$\overline{\lvert\boldsymbol{\theta}\rvert}$
Implementation St.Dev. standard	0.022(2)	0.138(7)	0.005(3)
Implementation St.Dev. $G = 0$	0.033(2)	0.191(9)	0.005(5)
Implementation St.Dev. $G = 0.25$	0.018(2)	0.110(6)	0.002(4)
Implementation St.Dev. $G = 1$	0.013(1)	0.065(4)	0.000(2)

Table 5.3: Estimated implementation-specific effects for the Gaussian mixture likelihood results shown in Table 5.2, calculated using the method described in Chapter 4.

perfect nested sampling. Dynamic nested sampling with $G = 1$ and $G = 0.25$ both reduce implementation-specific effects in all of the parameter estimation calculations as expected. However we are not able to measure a statistically significant difference in implementation-specific effects for $\log \mathcal{Z}$ with $G = 0$; this is because for evidence calculations implementation-specific effects represent a much smaller fraction of the total error.

The efficiency gains in Table 5.2 are slightly lower than those for the similar unimodal Gaussian likelihood (3.7) used in Table 5.1; this is because of the higher n_{init} value used, and because while implementation-specific effects are reduced by dynamic nested sampling they are not reduced by as large a factor as errors from the stochasticity of the nested sampling algorithm.

5.6 Conclusion

This chapter began with an analysis of the effects of changing the number of live points on the accuracy of nested sampling parameter estimation and evidence calculations. We then presented dynamic nested sampling (Algorithm 4), which varies the number of live points to allocate posterior samples efficiently for *a priori* unknown likelihoods and priors.

Dynamic nested sampling can be optimised specifically for parameter estimation, showing increases in computational efficiency over standard nested sampling (5.6) by

factors of up to 72 ± 5 in numerical tests. The algorithm can also increase evidence calculation accuracy, and can improve both evidence calculation and parameter estimation simultaneously. We discussed factors effecting the efficiency gain from dynamic nested sampling, including showing large improvements in parameter estimation are possible when the posterior mass is contained in a small region of the prior (as is typically the case in high-dimensional problems). Empirical tests show significant efficiency gains from dynamic nested sampling for a wide range likelihoods, priors, dimensions and estimators considered. Another advantage of dynamic nested sampling is that more accurate results can be obtained by continuing the run for longer, unlike in standard nested sampling. Finally we applied dynamic nested sampling software to a challenging multimodal posterior, with empirical tests showing that it gives similar performance gains to the unimodal cases and that it also reduced errors due to implementation-specific effects compared to standard nested sampling.

The many popular approaches and software implementations for standard nested sampling can be easily adapted for dynamic nested sampling, since it too only requires samples to be drawn randomly from the prior within some hard likelihood constraint. As a result, our new method can be used to increase computational efficiency while maintaining the strengths of standard nested sampling. Publicly available dynamic nested sampling packages include `dyPolyChord`, `dynesty` and `perfectns`.

Dynamic nested sampling has been applied to a variety of research problems in astrophysics, including astronomical image reconstruction (see Chapters 6 and 7), constraining the present day stellar mass function (Orazio et al., 2018), fitting light curves of transient sources (Guillochon et al., 2018) and mapping distances across the Perseus molecular cloud (Zucker et al., 2018).

Appendix 5.A Code

The code used to generate the numerical results and plots in this chapter is available at `https://github.com/ejhigson/dns`.

Appendix 5.B Estimating sampling errors in dynamic nested sampling

The technique for estimating sampling errors by resampling threads introduced in Chapter 3 can be applied to dynamic nested sampling runs with variable numbers of live points. Table 5.4 shows numerical tests of the bootstrap error estimates for dynamic nested sampling, calculated using the `nestcheck` package (Higson, 2018b). The results use $G = 1$ — this the most challenging case as most of the threads only cover part of the $\log X$ range explored by the run. The bootstrap error estimates match the sampling errors observed when the calculation is repeated many times, in agreement with the results for standard nested sampling in Chapter 3.

When n_{init} is low and $G = 1$, bootstrap replications may contain zero (or very few) threads which begin by sampling the whole prior. This typically does not matter for calculating parameter estimation errors as only the relative weights of points are used, but may lead to inaccurate estimates of evidence errors. In this case the threads from the initial exploratory run can be sampled separately (with replacement), ensuring every bootstrap replication contains n_{init} such threads — this approach was used for Table 5.4. When n_{init} is close to 1, estimates of $\log \mathcal{Z}$ uncertainties with this approach become imprecise, and the simulated weights method (see Chapter 3 for more details) may perform better.

Appendix 5.C Effect of varying the number of live points on evidence calculation accuracy

Nested sampling estimates the Bayesian evidence $\mathcal{Z}$ as the expectation of (2.19), as described in Section 2.4.1. The dominant source of uncertainty is the unknown shrinkage ratios t_i, which are independent random variables with probability density functions $P(t_i)$ given in (2.18). We now investigate the effect of increasing the number of live points n_i across some shrinkage t_i by considering (2.19) with all $t_{j\neq i}$ marginalised out and conditioned on t_i, defining

$$\mathcal{Z}(t_i) \equiv \int \left(\sum_j w_j(\boldsymbol{\theta}) \mathcal{L}_j \right) \prod_{j \neq i} P(t_j) \, \mathrm{d}t_j. \tag{5.10}$$

	$\log \mathcal{Z}$	$\overline{\theta_{\hat{1}}}$	median$(\theta_{\hat{1}})$
Mean result	-9.710(7)	0.0002(3)	0.0003(3)
Repeated runs St.Dev.	0.464(5)	0.0184(2)	0.0234(2)
Bootstrap St.Dev. / Repeats St.Dev.	0.99(1)	1.02(1)	1.00(1)
Bootstrap St.Dev. variation	17.1(2)%	6.07(6)%	11.5(1)%
Bootstrap C.I.$_{95\%}$	-8.94(2)	0.0304(8)	0.038(1)
Bootstrap $\pm$1St.Dev. coverage	67.7%	68.6%	68.4%
Bootstrap C.I.$_{95\%}$ coverage	95.6%	94.9%	94.7%
(continued)	C.I.$_{84\%}(\theta_{\hat{1}})$	$\overline{\lvert\boldsymbol{\theta}\rvert}$	median$(\lvert\boldsymbol{\theta}\rvert)$
Mean result	0.9904(4)	1.5890(3)	1.5316(3)
Repeated runs St.Dev.	0.0294(3)	0.0195(2)	0.0232(2)
Bootstrap St.Dev. / Repeats St.Dev.	1.03(1)	1.01(1)	1.00(1)
Bootstrap St.Dev. variation	13.1(1)%	6.69(7)%	10.9(1)%
Bootstrap C.I.$_{95\%}$	1.038(1)	1.6209(9)	1.569(1)
Bootstrap $\pm$1St.Dev. coverage	70%	68.5%	69.0%
Bootstrap C.I.$_{95\%}$ coverage	95.0%	95.2%	94.8%

Table 5.4: Bootstrap sampling error estimates for dynamic nested sampling of a 3-dimensional Gaussian likelihood (3.7) and a Gaussian prior (3.9) The table shows results from 5,000 dynamic nested sampling runs generated with `perfectns` using $G = 1$, $n_{\text{init}} = 20$ and with the same total number of samples as standard nested sampling with a constant $n = 200$ live points. The first two rows show the mean and standard deviation of the results of the $5,000$ calculations. The third row shows the mean of the error estimates from the bootstrap resampling technique for each run (using 200 replications), divided by the error observed from repeated calculations. The fourth row shows the standard deviations of bootstrap error estimates for single runs as a percentage of the mean estimate. The fifth row shows the mean of 500 bootstrap estimates of the one-tailed 95% credible interval on the calculation result given the sampling error, each using $1,000$ bootstrap replications. The final two rows show the empirical coverage of the bootstrap standard error and 95% credible interval from the $5,000$ repeated calculations. Numbers in brackets show the 1σ numerical uncertainty on the final digit.

For brevity in the remainder of this section we omit the explicit dependence of quantities such as point weights $w_i(\mathbf{t})$ on the shrinkage ratios $\mathbf{t}$.

For simplicity instead of using the trapezium rule we calculate point weight as

$$w_i = X_{i-1} - X_i = (1 - t_i) \prod_{k<i} t_k. \tag{5.11}$$

In this case uncertainty in t_i causes sampling errors in the weight of point i and all subsequent points[7] and

$$\sum_j w_j \mathcal{L}_j = \left[\sum_{j<i} w_j \mathcal{L}_j \right] + (1 - t_i) \left[\frac{w_i \mathcal{L}_i}{1 - t_i} \right] + t_i \left[\sum_{j>i} \frac{w_j \mathcal{L}_j}{t_i} \right], \tag{5.12}$$

where the terms in square brackets are independent of t_i. Substituting (5.12) into (5.10) and integrating gives

$$\mathcal{Z}(t_i) = \mathrm{E}[\mathcal{Z}_{<i}] + (1 - t_i)\mathrm{E}\left[\frac{\mathcal{L}_i w_i}{1 - t_i} \right] + t_i \mathrm{E}\left[\frac{\mathcal{Z}_{>i}}{t_i} \right], \tag{5.13}$$

where we have defined $\mathcal{Z}_{>i} \equiv \sum_{k>i} \mathcal{L}_k w_k$ and $\mathcal{Z}_{<i} \equiv \sum_{k<i} \mathcal{L}_k w_k$. The second term can be simplified by observing that as the shrinkage ratios are independent $\mathcal{L}_i w_i/(1 - t_i)$ is uncorrelated with $(1 - t_i)$, and that from (5.11) $\mathcal{L}_i w_i \propto (1 - t_i)$. Two uncorrelated random variables A and B must satisfy $\mathrm{E}[A] = \mathrm{E}[AB]/\mathrm{E}[B]$, so hence

$$\mathrm{E}\left[\frac{\mathcal{L}_i w_i}{1 - t_i} \right] = \frac{\mathrm{E}[\mathcal{L}_i w_i]}{\mathrm{E}[1 - t_i]} = \frac{\mathrm{E}[\mathcal{L}_i w_i]}{1 - \mathrm{E}[t_i]}. \tag{5.14}$$

Similarly $\mathcal{Z}_{>i}/t_i$ is uncorrelated with t_i and from (5.11) $\mathcal{Z}_{>i} \propto t_i$, so

$$\mathrm{E}\left[\frac{\mathcal{Z}_{>i}}{t_i} \right] = \frac{\mathrm{E}[\mathcal{Z}_{>i}]}{\mathrm{E}[t_i]}. \tag{5.15}$$

Hence (5.13) can be rewritten as

$$\mathcal{Z}(t_i) = \mathrm{E}[\mathcal{Z}_{<i}] + (1 - t_i)\frac{\mathrm{E}[\mathcal{L}_i w_i]}{(1 - \mathrm{E}[t_i])} + t_i \frac{\mathrm{E}[\mathcal{Z}_{>i}]}{\mathrm{E}[t_i]}. \tag{5.16}$$

Furthermore, from the distribution of the shrinkage ratios (2.18)

$$\mathrm{E}[t_i] = \frac{n_i}{1 + n_i}, \qquad \text{St.Dev.}[t_i] = \frac{{n_i}^{1/2}}{(n_i + 1)(n_i + 2)^{1/2}}. \tag{5.17}$$

[7] If the trapezium rule is used t_i also affects the weight of the previous point $i - 1$, but this has little effect on the results.

Substituting this into (5.16) gives

$$\mathcal{Z}(t_i) = \left(\mathrm{E}[\mathcal{Z}_{<i}] + \mathrm{E}[\mathcal{L}_i w_i](n_i+1)\right) + t_i\left(\frac{n_i+1}{n_i}\mathrm{E}[\mathcal{Z}_{>i}] - (1+n_i)\mathrm{E}[\mathcal{L}_i w_i]\right), \quad (5.18)$$

where terms in large brackets are independent of t_i. Using the expression for St.Dev.$[t_i]$ from (5.17), the standard deviation of $\mathcal{Z}(t_i)$ is

$$\text{St.Dev.}[\mathcal{Z}(t_i)] = \frac{1}{{n_i}^{1/2}(n_i+2)^{1/2}}\mathrm{E}[\mathcal{Z}_{>i}] - \frac{n_i^{1/2}}{(n_i+2)^{1/2}}\mathrm{E}[\mathcal{L}_i w_i]. \quad (5.19)$$

The expected number of samples (computational work) needed to increase the number of live points over some interval $(\mathcal{L}_a, \mathcal{L}_b)$ is proportional to the log prior shrinkage $\log X(\mathcal{L}_a) - \log X(\mathcal{L}_b)$. Hence the expected extra samples ΔN_s required to increase the local number of live points n_i is proportional to the interval $\log t_i$, which has an expected size of $1/n_i$. The change in the error on the evidence with extra samples is therefore

$$\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}N_\mathrm{s}}\text{St.Dev.}[\mathcal{Z}(t_i)] &= \frac{\mathrm{d}n_i}{\mathrm{d}N_\mathrm{s}}\frac{\mathrm{d}}{\mathrm{d}n_i}\text{St.Dev.}[\mathcal{Z}(t_i)] && (5.20)\\
&\propto n_i\frac{\mathrm{d}}{\mathrm{d}n_i}\text{St.Dev.}[\mathcal{Z}(t_i)] && (5.21)\\
&\propto -\frac{n_i+1}{{n_i}^{1/2}(n_i+2)^{3/2}}\mathrm{E}[\mathcal{Z}_{>i}] - \frac{n_i^{1/2}}{(n_i+2)^{3/2}}\mathrm{E}[\mathcal{L}_i w_i]. && (5.22)
\end{aligned}$$

This quantity can be easily calculated for a set of dead points with little computational cost. Typically $n_i \gg 2$, in which case the following relation approximately holds:

$$\frac{\mathrm{d}}{\mathrm{d}N_\mathrm{s}}\text{St.Dev.}[\mathcal{Z}(t_i)] \propto -\frac{\mathrm{E}[\mathcal{Z}_{\geq i}]}{n_i}, \quad (5.23)$$

where $\mathcal{Z}_{\geq i} \equiv \sum_{k\geq i}\mathcal{L}_k w_k(\mathbf{t})$. Thus the accuracy gained from taking additional samples is approximately proportional to the evidence contained in subsequent dead points. This makes sense as the dominant evidence errors are from statistically estimating shrinkages t_i which affect all subsequent points $j \geq i$.

Appendix 5.D Tuning for a specific parameter estimation problem

Dynamic nested sampling improves parameter estimation efficiency by placing more samples in $\log X$ regions with significant posterior mass and fewer in regions with little

posterior mass. However, for some likelihoods and parameter estimation problems a large contribution to errors comes from samples in $\log X$ regions containing extreme or highly variable parameter values but little posterior weight (see Figure 3.2 for a diagrammatic illustration). In this case the expression for sample importances (5.3) can be modified to favour points with parameter values which will have a large effect on the calculation.

For example, when estimating the global mean of some parameter or function of parameters $\mathrm{E}[f(\boldsymbol{\theta})] = \sum_i f(\boldsymbol{\theta}_i)\mathcal{L}_i w_i$, one could place additional weight on regions with parameter values that have a large effect on results by calculating importances as

$$I_{\mathrm{param}}(i) \propto |f(\boldsymbol{\theta}_i) - \mathrm{E}[f(\boldsymbol{\theta})]|\, \mathcal{L}_i w_i. \tag{5.24}$$

This expression is highly variable as each point i is a single sample from an iso-likelihood contour $\mathcal{L}(\boldsymbol{\theta}) = \mathcal{L}_i$ which may cover a wide range of parameters. However dynamic nested sampling (Algorithm 4) uses only the first and last points of high importance in allocating new threads, so (5.24) captures $\log X$ regions in which some samples have extreme or highly variable parameter values. When tuning dynamic nested sampling for calculating the mean of a parameter $\theta_{\hat{1}}$, (5.24) becomes

$$I_{\mathrm{param}}(i) \propto \left|\theta_{i,\hat{1}} - \overline{\theta_{\hat{1}}}\right| \mathcal{L}_i w_i, \tag{5.25}$$

where $\overline{\theta_{\hat{1}}}$ is the global mean of $\theta_{\hat{1}}$ and $\theta_{i,\hat{1}}$ is the i^{th} sample's $\theta_{\hat{1}}$ value.

We illustrate tuning for a specific parameter by using dynamic nested sampling with a d-dimensional spherical unit Cauchy likelihood

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{\Gamma(\frac{1+d}{2})}{\pi^{(d+1)/2}}\left(1 + |\boldsymbol{\theta}|^2\right)^{-(\frac{d+1}{2})}. \tag{3.8 revisited}$$

The Cauchy likelihoods have extremely heavy tails and (except in high dimensions) have significant posterior mass present across almost the entire range of $\log X$ explored, as shown in Figure 5.10. We therefore expect relatively low efficiency gains for dynamic parameter estimation ($G = 1$) in this case, but use it for a proof of principle.

For a Cauchy likelihood (3.8) with a co-centred spherically symmetric uniform prior, the analytic value of $\mathrm{E}[\theta_{\hat{1}}]$ is 0 and each iso-likelihood contour $\mathcal{L}(\boldsymbol{\theta}) = \mathcal{L}(X)$ is a spherically symmetric surface with radius $|\boldsymbol{\theta}|$. The expectation of $|\theta_{\hat{i}}|$ on such an iso-likelihood contour is $|\boldsymbol{\theta}|/\sqrt{d}$, so the analytical expectation of the importance (5.25) is

$$I_{\mathrm{param}}(X) \propto |\boldsymbol{\theta}| X \mathcal{L}(X)/\sqrt{d}. \tag{5.26}$$

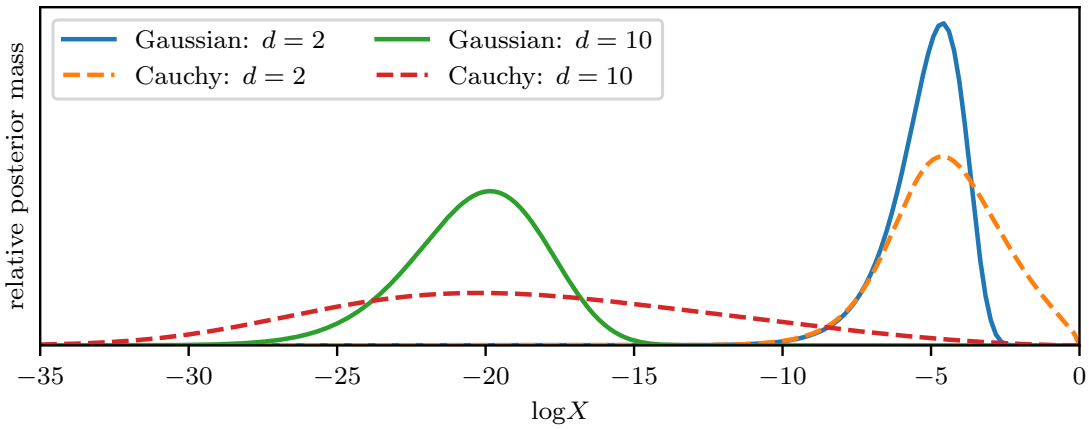


Figure 5.10: Relative posterior mass ($\propto \mathcal{L}(X)X$) as a function of $\log X$ for Cauchy likelihoods (3.8), with Gaussian likelihoods (3.7) shown for comparison. Each has a Gaussian prior (3.9) with $\sigma_\pi = 10$. The lines are scaled so that the area under each of them is equal.

Figure 5.11 shows the allocation of live points by dynamic nested sampling with and without tuning. The numbers of live points as a function of $\log X$ for the tuned runs are consistent with (5.26), showing that samples can be allocated accurately when the tuned importance function is used.

Table 5.5 shows the efficiency gain for dynamic nested sampling for a 10-dimensional Cauchy likelihood (3.8) with a Gaussian prior (3.9) and $\sigma_\pi = 10$. When estimating $\overline{\theta_{\hat{1}}}$ the calculation is dominated by samples in the tails of the distribution with low likelihoods. As a result, compared to standard nested sampling, dynamic nested sampling with $G = 1$ slightly increases the variation of results — giving an efficiency gain (5.6) of less than 1. Tuned dynamic nested sampling is able to improve the efficiency gain for $\overline{\theta_{\hat{1}}}$, as shown in the final row of Table 5.5, although for the Cauchy likelihood the resulting gain is still small. Using the tuned importance function affects the performance gain for other quantities — for example in this case it significantly improves estimates of the second moment of the distribution $\overline{\theta^2_{\hat{1}}}$ in comparison to the $G = 1$ case without tuning, but reduces the accuracy of estimates of the 84% credible interval of $\theta_{\hat{1}}$.

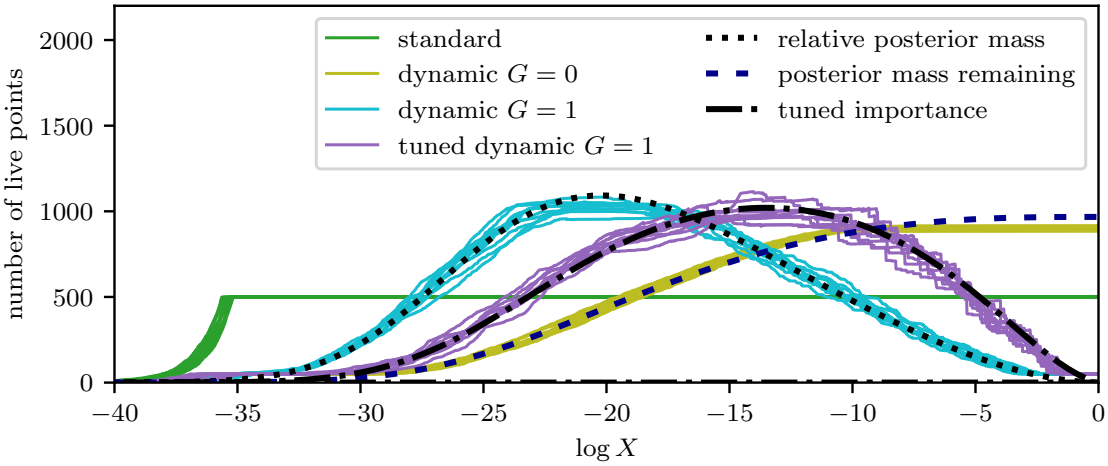


Figure 5.11: Live point allocation for a 10-dimensional Cauchy likelihood (3.8) with a Gaussian prior (3.9) and $\sigma_\pi = 10$. Solid green lines show the number of live points as a function of $\log X$ for 10 standard nested sampling runs. Solid yellow, blue and purple lines show 10 dynamic nested sampling runs with $G = 0$, $G = 1$ and $G = 1$ with a tuned importance function (5.25) respectively. Dynamic runs use a similar number of samples to standard runs. The dotted, dashed and dot-and-dash lines show the relative posterior mass $\propto \mathcal{L}(X)X$, the posterior mass remaining $\propto \int_{-\infty}^{X} \mathcal{L}(X')X' \,\mathrm{d}X'$ and the analytical expectation of the tuned importance function (5.26). For comparison these lines are scaled to have the same area under them as the average of the number of live point lines.

Appendix 5.E Additional numerical tests

5.E.1 Exponential power likelihoods

This section contains additional tests of dynamic nested sampling using 10-dimensional exponential power likelihoods (5.5) with $b = 2$ and $b = \frac{3}{4}$; compared to Gaussian likelihoods (3.7) these have lighter and heavier tails respectively. As in Section 5.4, each test uses a Gaussian prior (3.7) with $\sigma_\pi = 10$.

Figures 5.12 and 5.13 show that the dynamic nested sampling algorithm can accurately and consistently allocate live points for these likelihoods. Tables 5.6 and 5.7 show the reduction in errors from dynamic nested sampling compared to standard nested sampling in these two cases, as measured by repeated calculations. This corresponds to increases in efficiency (5.6) for evidence calculation ($G = 0$) and parameter estimation ($G = 1$) by factors of 1.25 ± 0.04 and up to 6.8 ± 0.2 respectively in the $b = 2$ case, and by factors of 1.62 ± 0.05 and up to 3.11 ± 0.09 in the $b = \frac{3}{4}$ case.

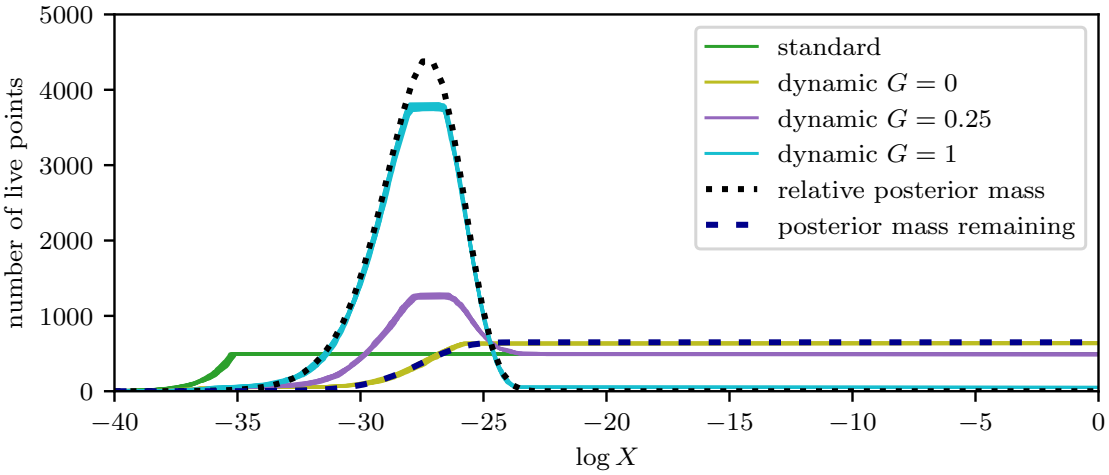


Figure 5.12: As in Figure 5.3 but with a 10-dimensional exponential power likelihood (5.5) with $b = 2$.

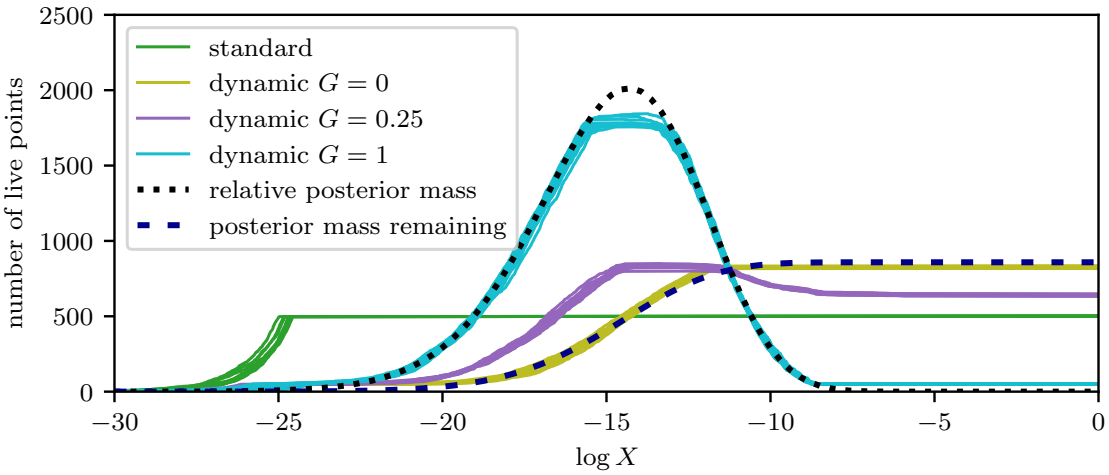


Figure 5.13: As in Figure 5.3 but with a 10-dimensional exponential power likelihood (5.5) with $b = \frac{3}{4}$.

	samples	$\log \mathcal{Z}$	$\overline{\theta_{\hat{1}}}$	$\overline{\theta_{\hat{1}}^2}$
St.Dev. standard	18,209	0.167(4)	0.0124(3)	0.238(5)
St.Dev. $G = 0$	18,165	0.133(3)	0.0119(3)	0.214(5)
St.Dev. $G = 1$	18,181	0.320(7)	0.0128(3)	0.236(5)
St.Dev. $G = 1$ tuned	18,181	0.244(5)	0.0106(2)	0.185(4)
Gain $G = 0$		1.6(1)	1.08(7)	1.23(8)
Gain $G = 1$		0.27(2)	0.94(6)	1.01(6)
Gain $G = 1$ tuned		0.46(3)	1.35(9)	1.6(1)
(continued)	C.I.$_{84\%}(\theta_{\hat{1}})$	$\overline{\lvert\boldsymbol{\theta}\rvert}$	median($\lvert\boldsymbol{\theta}\rvert$)	
St.Dev. standard	0.055(1)	0.180(4)	0.165(4)	
St.Dev. $G = 0$	0.056(1)	0.173(4)	0.165(4)	
St.Dev. $G = 1$	0.044(1)	0.157(4)	0.125(3)	
St.Dev. $G = 1$ tuned	0.045(1)	0.141(3)	0.130(3)	
Gain $G = 0$	0.97(6)	1.08(7)	0.99(6)	
Gain $G = 1$	1.6(1)	1.32(8)	1.7(1)	
Gain $G = 1$ tuned	1.5(1)	1.6(1)	1.6(1)	

Table 5.5: Test of tuned dynamic nested sampling with a 10-dimensional Cauchy likelihood (3.8), and a Gaussian prior (3.9) with $\sigma_\pi = 10$. The first four rows show the standard deviation of $1,000$ calculations for standard nested sampling and dynamic nested sampling with $G = 0$, $G = 1$ and with a tuned importance function (5.25) and $G = 1$. The final three rows show the computational efficiency gain (5.6) from dynamic nested sampling over standard nested sampling in each case. The first column shows the mean number of samples for the $1,000$ runs. The remaining columns show calculations of the log evidence, the mean, second moment and 84% one-tailed credible interval of the parameter $\theta_{\hat{1}}$, and the mean and median radial coordinate $|\boldsymbol{\theta}|$. Numbers in brackets show the 1σ numerical uncertainty on the final digit.

5.E.2 Gaussian mixture likelihoods

Table 5.8 shows comparisons of dynamic nested sampling results with analytically calculated values for the Gaussian mixture likelihood (5.8) with a Gaussian prior (3.9). The mean results are very close to the correct values, showing that there is no significant sampling bias. As a result the root-mean-squared-errors and standard deviations are almost identical, meaning efficiency gain estimates from (5.6) can be used reliably (as for perfect nested sampling).

	samples	$\log \mathcal{Z}$	$\overline{\theta_{\hat{1}}}$	median($\theta_{\hat{1}}$)
St.Dev. standard	18,093	0.228(2)	0.00870(9)	0.0110(1)
St.Dev. $G = 0$	18,052	0.204(2)	0.0107(1)	0.0147(1)
St.Dev. $G = 0.25$	18,056	0.228(2)	0.00587(6)	0.00777(8)
St.Dev. $G = 1$	18,058	0.686(7)	0.00363(4)	0.00471(5)
Gain $G = 0$		1.25(4)	0.66(2)	0.56(2)
Gain $G = 0.25$		1.00(3)	2.20(6)	2.01(6)
Gain $G = 1$		0.110(3)	5.7(2)	5.5(2)
(continued)	C.I.$_{84\%}(\theta_{\hat{1}})$	$\overline{\lvert\boldsymbol{\theta}\rvert}$	median($\lvert\boldsymbol{\theta}\rvert$)	
St.Dev. standard	0.0133(1)	0.00809(8)	0.0102(1)	
St.Dev. $G = 0$	0.0169(2)	0.00917(9)	0.0108(1)	
St.Dev. $G = 0.25$	0.00906(9)	0.00547(5)	0.00654(7)	
St.Dev. $G = 1$	0.00549(5)	0.00338(3)	0.00391(4)	
Gain $G = 0$	0.62(2)	0.78(2)	0.88(2)	
Gain $G = 0.25$	2.14(6)	2.19(6)	2.41(7)	
Gain $G = 1$	5.8(2)	5.7(2)	6.8(2)	

Table 5.6: As in Table 5.1 but with a 10-dimensional exponential power likelihood (5.5) with $b = 2$

	samples	$\log \mathcal{Z}$	$\overline{\theta_{\hat{1}}}$	median($\theta_{\hat{1}}$)
St.Dev. standard	12,855	0.157(2)	0.0261(3)	0.0320(3)
St.Dev. $G = 0$	12,824	0.123(1)	0.0283(3)	0.0391(4)
St.Dev. $G = 0.25$	12,827	0.138(1)	0.0222(2)	0.0289(3)
St.Dev. $G = 1$	12,833	0.432(4)	0.0160(2)	0.0194(2)
Gain $G = 0$		1.62(5)	0.85(2)	0.67(2)
Gain $G = 0.25$		1.30(4)	1.39(4)	1.22(3)
Gain $G = 1$		0.132(4)	2.66(8)	2.70(8)
(continued)	C.I.$_{84\%}(\theta_{\hat{1}})$	$\overline{\lvert\boldsymbol{\theta}\rvert}$	median($\lvert\boldsymbol{\theta}\rvert$)	
St.Dev. standard	0.0439(4)	0.0545(5)	0.0657(7)	
St.Dev. $G = 0$	0.0487(5)	0.0574(6)	0.0651(7)	
St.Dev. $G = 0.25$	0.0374(4)	0.0454(5)	0.0522(5)	
St.Dev. $G = 1$	0.0266(3)	0.0342(3)	0.0372(4)	
Gain $G = 0$	0.81(2)	0.90(3)	1.02(3)	
Gain $G = 0.25$	1.38(4)	1.44(4)	1.58(4)	
Gain $G = 1$	2.71(8)	2.54(7)	3.11(9)	

Table 5.7: As in Table 5.1 but with a 10-dimensional exponential power likelihood (5.5) with $b = \frac{3}{4}$

	$\log \mathcal{Z}$	$\overline{\theta_{\hat{1}}}$	$\overline{\theta_{\hat{2}}}$	$\overline{\theta_{\hat{3}}}$	$\overline{\theta_{\hat{4}}}$
Analytic values	-32.3442	0.3980	0.3980	0	0
Mean standard	-32.351(8)	0.397(3)	0.388(6)	0.0012(7)	-0.0004(7)
Mean $G = 0$	-32.352(7)	0.393(3)	0.380(8)	0.0011(8)	-0.0002(8)
Mean $G = 0.25$	-32.336(8)	0.397(2)	0.386(5)	-0.0001(6)	-0.0007(5)
Mean $G = 1$	-32.34(2)	0.399(1)	0.385(3)	0.0003(4)	-0.0004(4)
St.Dev. standard	0.181(6)	0.057(2)	0.126(4)	0.0146(5)	0.0161(5)
St.Dev. $G = 0$	0.160(5)	0.076(2)	0.176(6)	0.0182(6)	0.0178(6)
St.Dev. $G = 0.25$	0.170(5)	0.046(1)	0.105(3)	0.0134(4)	0.0123(4)
St.Dev. $G = 1$	0.36(1)	0.032(1)	0.069(2)	0.0087(3)	0.0089(3)
RMSE standard	0.181(6)	0.057(2)	0.127(4)	0.0147(4)	0.0161(5)
RMSE $G = 0$	0.160(5)	0.076(3)	0.177(6)	0.0182(6)	0.0178(6)
RMSE $G = 0.25$	0.170(5)	0.046(1)	0.106(3)	0.0134(5)	0.0123(4)
RMSE $G = 1$	0.36(1)	0.032(1)	0.070(2)	0.0086(3)	0.0089(3)
St.Dev. gain $G = 0$	1.3(1)	0.56(5)	0.51(5)	0.64(6)	0.82(7)
St.Dev. gain $G = 0.25$	1.1(1)	1.5(1)	1.5(1)	1.2(1)	1.7(2)
St.Dev. gain $G = 1$	0.25(2)	3.3(3)	3.4(3)	2.9(3)	3.3(3)
RMSE gain $G = 0$	1.3(1)	0.56(6)	0.51(5)	0.65(6)	0.82(7)
RMSE gain $G = 0.25$	1.1(1)	1.5(1)	1.4(1)	1.2(1)	1.7(2)
RMSE gain $G = 1$	0.25(2)	3.3(3)	3.3(3)	2.9(3)	3.3(3)

Table 5.8: Comparison of results from the nested sampling runs used in Table 5.2 with analytically calculated values for different quantities (shown in the first row). The next 12 rows show mean, the standard deviation and root mean squared errors for the standard nested sampling runs and the dynamic nested sampling runs with $G = 0$, $G = 0.25$ and $G = 1$. The final 6 rows show efficiency gains calculated with the standard deviation as in (5.6), and using the root-mean-squared-error instead of the standard deviation. Columns show calculations of the log evidence and the mean of the first 4 parameters. The mean dynamic nested sampling results agree closely with the analytic values, indicating that there is no significant sampling bias. Numbers in brackets show the 1σ numerical uncertainty on the final digit.

Appendix 5.F Dynamic nested sampling without repeatedly restarting runs

This section describes the alternative dynamic nested sampling algorithm used by `dyPolyChord` to avoid frequent resuming of the nested sampling process part way through the run. After the initial exploratory run with n_{init} live points, an allocation of live points which varies with likelihood $n(\mathcal{L})$ is calculated and used to generate all the remaining samples in a single run. The number of live points is increased during the run by sampling more than one live point from within a given iso-likelihood contour, and reduced by not replacing dead points when they are removed. The user must specify the approximate total number of samples to be taken, N_{total}, either as a constant or a function of the number of samples taken by the initial run N_{init}.

The target number of live points $n(\mathcal{L})$ is calculated using importances (5.4) of the dead points in the initial run; as the number of live points n_{init} is constant, the samples are evenly distributed in $\log X$ and the point importances are proportional to the importances of each $\log X$ region. $n(\mathcal{L})$ is calculated piecewise at each point i as

$$n(\mathcal{L}_i) = \begin{cases} K\, I(G,i) - n_{\text{init}} & \text{if } K\, I(G,i) > n_{\text{init}}, \\ 0 & \text{otherwise,} \end{cases} \tag{5.27}$$

where $I(i,G)$ is point i's relative importance, and each $n(\mathcal{L}_i)$ rounded to the nearest integer. The constant K is chosen so that approximately the right number of samples is taken — i.e. so that $n(\mathcal{L})$ satisfies

$$\int n(\mathcal{L}) \frac{\mathrm{d}\log X(\mathcal{L})}{\mathrm{d}\mathcal{L}}\, \mathrm{d}\mathcal{L} \approx N_{\text{total}} - N_{\text{init}}. \tag{5.28}$$

If $N_{\text{total}} \gg N_{\text{init}}$, (5.27) allocates live points approximately in proportion to the importances calculated from the initial run. Otherwise $n(\mathcal{L})$ is only non-zero in the region of high importance (where $I > n_{\text{init}}/K$), and will result in approximately equal sample weights in this region in the final combined run with lower weights elsewhere. Given the samples already taken by the initial exploratory run and the number of remaining samples available $N_{\text{total}} - N_{\text{init}}$, (5.27) approximately maximises the information content (Shannon entropy of the samples) (5.1). In practice estimates of $n(\mathcal{L})$ from (5.27) contain random noise from the stochasticity of the nested sampling algorithm. For better results the piecewise importance function can be smoothed before calculating $n(\mathcal{L})$;

by default `dyPolyChord` uses a Savitzky-Golay filter (Savitzky and Golay, 1964) with polynomial order 3 and window size $2n_{\text{init}} + 1$.

This procedure is set out more formally in Algorithm 5; for an example implementation see the `dyPolyChord` package and its documentation.

Output : Samples and live points information **n**.
Input : Goal G, n_{init}, approximate number of samples to take N_{total}.

Generate an initial nested sampling run with a constant number of live points n_{init};
calculate $n(\mathcal{L})$ from (5.27) using point importances $I(G, i)$ and the number of samples in the initial run N_{init};
perform nested sampling run with $n(\mathcal{L})$ live points, beginning by resuming initial the run at the first point where $n(\mathcal{L}_i) > 0$ and terminating[8] after the last point where $n(\mathcal{L}_i) > 0$;
merge the nested sampling runs generated and return the combined run.

Algorithm 5: The alternative dynamic nested sampling algorithm used by `dyPolyChord`.

[8] In principle $n(\mathcal{L})$ may drop to zero then, at some larger likelihood, become non-zero again — although this is very unlikely in practice. In this case the run can terminate when $n(\mathcal{L}) = 0$, then be restarted at the higher likelihood when $n(\mathcal{L})$ is again non-zero by resuming the initial exploratory run at this later point.

Chapter 6

Bayesian sparse reconstruction

This chapter presents a principled Bayesian framework for signal reconstruction, in which the signal is modelled by basis functions whose number (and form, if required) is determined by the data themselves. Furthermore, by using a product-space approach, the number and type of basis functions can be treated as integer parameters and their posterior distributions sampled directly. We show that order-of-magnitude increases in computational efficiency are possible from this technique compared to calculating the Bayesian evidences separately, and that further computational gains are possible using it in combination with dynamic nested sampling. We demonstrate our method for noisy 1- and 2-dimensional signals, including astronomical images. This chapter is an edited version of the first part of Higson et al. (2019c).

6.1 Introduction

Sparse signal processing and Bayesian inference are both well-established methods for data analysis, and have a considerable amount in common. However, these two approaches are often considered somewhat distinct from one another, and this is often reflected in the relatively small overlap of the communities who develop and apply each technique. Nevertheless Bayesian interpretations of sparse signal processing techniques have been pursued by a number of authors in the signal processing community — for example sparsity-promoting Bayesian approaches to compressed sensing, regression and classification, and basis selection can be found in Ji et al. (2008), Tipping (2001) and Wipf and Rao (2004) respectively. In addition, Bayesian inference with imposed spar-

sity has been applied to a variety of astronomical problems. These include inferring the temperature structure of the solar corona (Warren et al., 2017), estimating photometric redshifts of blended sources (Jones and Heavens, 2018), and imaging solar flares by representing them as a collection of geometric shapes (Sciacchitano et al., 2019).

In this chapter we outline a principled Bayesian approach for simultaneously imposing sparsity and performing dictionary learning to determine the optimal basis set for representing the signal, and discuss how Bayesian inference provides a very natural framework for sparsity. In our method a signal is modelled as the superposition of a set of basis functions, whose number and form are determined by the data themselves. Sparsity can be imposed directly via the prior on the number of basis functions N, while simultaneous dictionary learning is performed through the estimation of parameters describing the location and shape of the basis functions.

The optimum number of basis functions N with which to model a signal can be determined using Bayesian model selection by calculating the Bayesian evidence for each value. However it is equivalent (and often more computationally efficient and convenient) to treat N as an integer parameter, and sample directly from the joint posterior of N and the other parameters describing the N basis functions. The final inference may then be obtained by either by choosing the maximum *a posteriori* value of N or, better, by marginalising over N to give a multi-model solution (Parkinson and Liddle, 2013) with the fit for each number of basis functions weighted by its posterior probability. This method can be further generalised to select from a variety of types of basis functions T (such as Gaussians, Fourier modes, wavelet families, shapelets, etc.), with the full version involving inference over the joint space of T, N and the basis functions' parameters.

While our principled approach is computationally expensive, we show that it is practical in the low data regime using current numerical methods at reasonable computational cost (see Table 6.2 in Appendix 6.B for details of the number of core hours used to produce our results). In addition, this chapter is intended as a proof of principle for applications where our method is not currently feasible but will be made so in the future by advances in numerical methods and increases in computational power.

The chapter proceeds as follows: Section 6.2 describes standard regression techniques, regularisation and sparsity. Section 6.3 then provides a Bayesian perspective on these topics — including introducing our formulation of "Bayesian sparse recon-

struction" and a discussion of how it can be implemented numerically. Sections 6.4 and 6.5 demonstrate applying our approach to 1- and 2-dimensional signal processing, including of astronomical images from the Hubble Space Telescope eXtreme Deep Field (Illingworth et al., 2013).

6.2 Regression, regularisation and sparsity

We begin by presenting some background on standard approaches to regression, regularisation and sparsity. This provides context for the Bayesian framework presented in Section 6.3, in which all these methods may be reinterpreted. This section is intended to draw out the common themes in numerous popular signal reconstruction methods, and describe them in a unified manner.

6.2.1 Standard non-parametric regression

Regression involves using data points $\{\boldsymbol{x}_d, y_d\}$ (including random noise) to reconstruct some function $y = f(\boldsymbol{x}; \boldsymbol{\theta})$, where $\boldsymbol{\theta}$ is some number of free parameters and the semicolon separates variables from parameters. Such inverse problems are common in science, and are typically ill-posed[1]. For example, in monochrome image reconstruction each data point is a pixel with 2-dimensional (centre) position $\boldsymbol{x}$ and scalar intensity value y. In general $\boldsymbol{x}$ and y can be vectors of any dimension, but for simplicity in this chapter we consider only scalar outputs y. The results easily generalise to vector outputs.

When a good model for the data is not available *a priori*, a traditional non-parametric approach is to use a *free-form* solution (Sivia and Skilling, 2006) in which the function is pixelated and the value at each pixel is fitted. This is a standard way of performing "brute-force" numerical calculations on computers, and is equivalent to fitting a delta function (or more accurately "top-hat") basis function centred on each of the M pixels with their amplitudes as free parameters, giving M degrees of freedom. Ironically, such "non-parametric" approaches thus contain many parameters — typically far more than "parametric" approaches. The free form approach is illustrated for 1-dimensional input x in Figure 6.1 (which is based on Figure 6.1 of Sivia and Skilling, 2006), but can be performed in arbitrary dimensions.

[1] An "ill-posed" problem does not satisfy all three of the conditions for a problem to be "well-posed" outlined by Hadamard (1902). The conditions are: a solution exists, the solution is unique and the

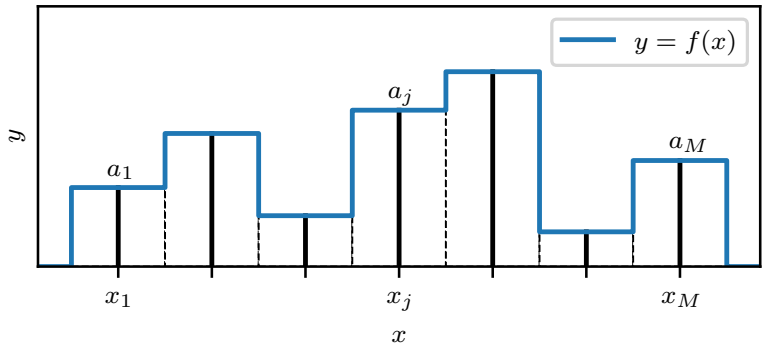


Figure 6.1: Free-form decomposition of a 1-dimensional function $y = f(x)$ into M pixels with amplitudes (free parameters) $\boldsymbol{\theta} = (a_1, a_2, \ldots, a_M)$.

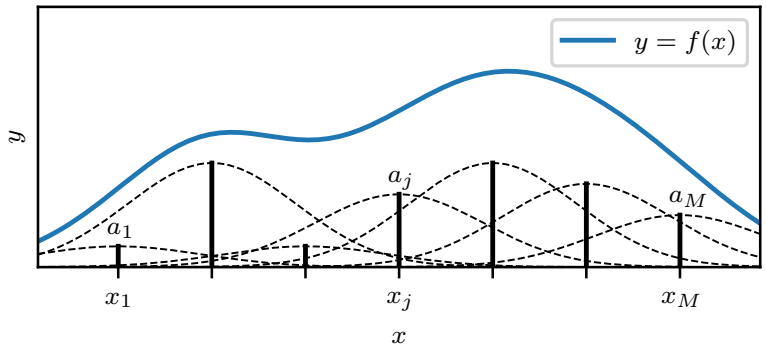


Figure 6.2: Free-form decomposition of a 1-dimensional function $y = f(x)$ into M Gaussian basis functions with standard deviation σ, each centred on a pixel, with amplitudes (free parameters) $\boldsymbol{\theta} = (a_1, a_2, \ldots, a_M)$.

Smoothness can be encoded into the solution by replacing the delta functions with broader basis functions $\phi(\boldsymbol{x}; \boldsymbol{x}_j, \sigma)$, with fixed centres $\boldsymbol{x}_j$ located on each of the M pixels and their width determined by a shared shape parameter σ. For Gaussian basis functions:

$$f(\boldsymbol{x}; \boldsymbol{a}, \sigma) = \sum_{j=1}^{M} a_j \phi(\boldsymbol{x}; \boldsymbol{x}_j, \sigma) = \sum_{j=1}^{M} a_j \exp\left(-\frac{|\boldsymbol{x} - \boldsymbol{x}_j|^2}{2\sigma^2}\right). \tag{6.1}$$

This is illustrated for a 1-dimensional input x in Figure 6.2 (based on Figure 6.7 of Sivia and Skilling, 2006).

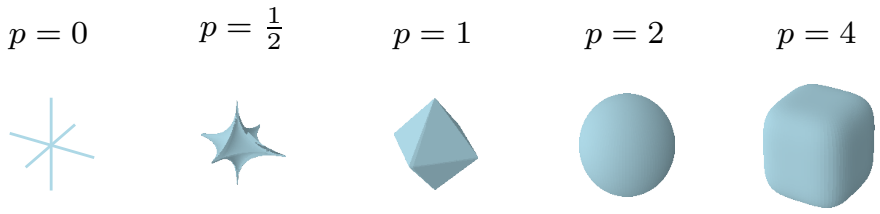


Figure 6.3: $L_p = 1$ surfaces in 3 dimensions for different values of p, with L_0 representing the number of non-zero components of the vector.

6.2.2 Optimisation and regularisation

The values of the parameters $\boldsymbol{\theta}$ are typically chosen by minimising the squared L_2 norm of the differences between the model and the data (also referred to as the squared residuals or χ^2). Here the L_p norm of a vector is defined for $p > 0$ as $\|\boldsymbol{v}\|_p \equiv (\sum_i |v_i|^p)^{1/p}$ and the L_0 norm is the number of non-zero components; this is illustrated for different values of p in Figure 6.3. The squared L_2 norm approach yields the maximum likelihood estimate (MLE) for $\boldsymbol{\theta}$ under certain restrictive conditions,[2] although it is commonly applied when these are not met; see Sivia and Skilling (2006, Chapter 8) for a more detailed discussion and recommended modifications to the least squares procedure for different types of data.

When using the squared L_2 norm, the optimisation is

$$\min_{\boldsymbol{\theta}} \sum_{d=1}^{D} \left(y_d - f(\boldsymbol{x}_d; \boldsymbol{\theta})\right)^2 = \min_{\boldsymbol{\theta}} \|\boldsymbol{y} - \hat{\boldsymbol{y}}\|_2^2, \tag{6.2}$$

where $\boldsymbol{y} = \{y_1, \ldots, y_D\}$ are the data values and $\hat{\boldsymbol{y}} = \{f(\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_D)\}$ are the fit values. For simplicity we assume for the moment that the shape of the basis functions is fixed and only the amplitudes are free parameters, in which case

$$\min_{\boldsymbol{\theta}} \|\boldsymbol{y} - \hat{\boldsymbol{y}}\|_2^2 = \min_{\boldsymbol{a} \in \mathcal{R}^M} \|\boldsymbol{y} - \Phi \boldsymbol{a}\|_2^2, \tag{6.3}$$

where the vector $\boldsymbol{a} = (a_1, a_2, \ldots, a_M)$ determines the basis functions' amplitudes and $\Phi = (\phi_1, \phi_2, \ldots, \phi_M)$ is a $D \times M$ basis matrix.

Typically a regularisation term is added to penalise more complex models; this is to prevent the analysis fitting noise in the data set and producing a result which will not generalise to new data sets ("overfitting"). Some popular choices are:

behaviour of the solution changes continuously with changes in the parameters and data.

[2] These include that the residuals on each data point must be independently normally distributed, and that there are no errors in the independent variables $\boldsymbol{x}$.

- The (squared) L_2 norm — used in the Wiener filter (Wiener, 1949) and ridge regression (Hoerl and Kennard, 1970):

$$\min_{\boldsymbol{a}\in\mathcal{R}^M} \|\boldsymbol{y} - \Phi\boldsymbol{a}\|_2^2 + \lambda\|\boldsymbol{a}\|_2^2. \tag{6.4}$$

- The L_1 norm — used in the Lasso (Tibshirani, 1996), compressed sensing and for imposing sparsity:

$$\min_{\boldsymbol{a}\in\mathcal{R}^M} \|\boldsymbol{y} - \Phi\boldsymbol{a}\|_2^2 + \lambda\|\boldsymbol{a}\|_1. \tag{6.5}$$

- The L_0 norm — used in matching pursuit (Mallat and Zhang, 1993), iterative thresholding (Elad et al., 2007), compressed sensing and for imposing sparsity:

$$\min_{\boldsymbol{a}\in\mathcal{R}^M} \|\boldsymbol{y} - \Phi\boldsymbol{a}\|_2^2 + \lambda\|\boldsymbol{a}\|_0, \tag{6.6}$$

  where $\|\boldsymbol{a}\|_0$ simply counts the number of non-zero elements in the amplitude vector $\boldsymbol{a}$.

- The entropy — used in the maximum entropy method (MEM):

$$\min_{\boldsymbol{a}\in\mathcal{R}^M} \|\boldsymbol{y} - \Phi\boldsymbol{a}\|_2^2 - \lambda S(\boldsymbol{a}), \tag{6.7}$$

$$S(\boldsymbol{a}) = \sum_{i=1}^{M} a_i - m_i - a_i \ln\left(\frac{a_i}{m_i}\right), \tag{6.8}$$

  where m_i is a (model) amplitude value assigned to each basis function (Ables, 1974; Gull and Daniell, 1978).

In principle, such optimisations define a "solution curve" $\hat{\boldsymbol{a}}(\lambda)$. To obtain a particular solution one must choose a value for the regularisation parameter λ, which determines the relative importance of the accuracy of the fit to the data and the value of the regularising function; it is often chosen *a priori* but can be determined using heuristics or cross-validation. For example, the regularisation constant for MEM has historically been chosen so the residual statistic equals its expectation value — i.e. so $\chi^2 = D$ where D is the number of data points (Sivia and Skilling, 2006). A more modern approach is to choose the value of λ which maximises the Bayesian evidence; this can also be used to select quantities such as the width σ of the basis functions shown in Figure 6.2 (Sivia and Skilling, 2006).

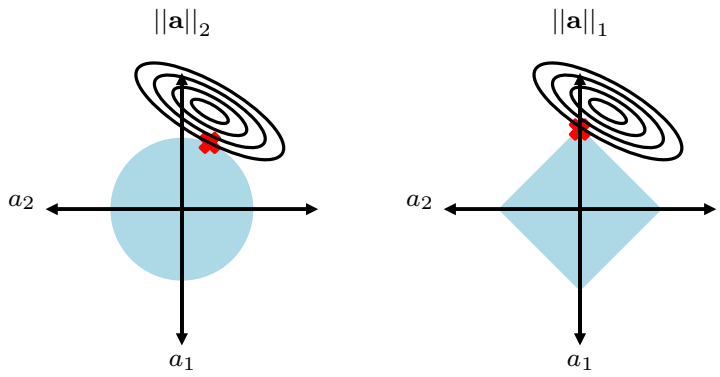


Figure 6.4: Illustration of L_p norms promoting sparsity when $p < 2$. The blue circle on the left plot shows the region of the parameter space (a_1, a_2) with an L_2 norm less than some maximum value, and the blue diamond on the right plot shows the region with an L_1 norm less than some maximum value. The black contours on each plot show an objective function to be optimised. Due to its angular shape, the region constrained by a maximum L_1 norm is more likely to have its maximum value of the objective function at a coordinate where one of the parameters is zero than the L_2 region.

It is worth noting that Equations (6.4) to (6.7) refer to the *synthesis* formulation which optimises over the parameters $\boldsymbol{a}$. An alternative is the *analysis* approach in which the optimisation is performed directly with respect to the (vectorised) function $f(\boldsymbol{x})$, and $\boldsymbol{a}$ is replaced in Equations (6.4) to (6.7) with $\Phi^{-1} f(\boldsymbol{x})$. This technique is commonly used in radio interferometry — see for example Maisinger et al. (2004), McEwen and Wiaux (2011) and Cai et al. (2018).

6.2.3 Sparse representations

In many practical signal and image processing applications we can use prior knowledge that the physical signals have "sparse" representations in which they have very few non-zero components (a low L_0 norm). For example, astronomical images with many pixels can often be well represented by a relatively small number of point sources or wavelets. Sparse solutions are promoted by choosing a regularisation term L_p with $p < 2$, in which case L_p surfaces have singular points at sparse solutions (Bach et al., 2012); this is illustrated graphically in Figure 6.4.

Sparsity is key to *compressed sensing* (Candès et al., 2006a,b; Donoho, 2006): a popular signal processing technique for efficiently recovering high-dimensional vector signals under the assumption that they are sparse in some basis (see Eldar and Kutyniok,

2012, for an introduction). Sparse solutions can be found by L_0-optimisation, but this is computationally challenging and is non-convex, meaning standard convex optimisation cannot be used. The success of compressed sensing is based on instead using the L_1-norm — the smallest p for which the L_p norm is convex. Compressed sensing theory shows that in some cases the L_1-norm can give an identical solution to the L_0-norm, and that in other cases the difference between the solutions is bounded. Compressed sensing has been applied successfully to a variety astronomical problems; see for example Bobin et al. (2008) and Wiaux et al. (2009).

6.2.4 Adaptive basis functions and dictionary learning

In order to find representations for data sets which are sparse (use relatively few basis functions) we now generalise the reconstructions described in (6.2.1) by allowing each basis function's location and shape to be determined by parameters $\boldsymbol{p}_i$ and fitted to the data. The signal is reconstructed as

$$f(\boldsymbol{x}; \boldsymbol{a}, \boldsymbol{p}_1, \ldots, \boldsymbol{p}_N) = \sum_{i=1}^{N} a_i \phi(\boldsymbol{x}; \boldsymbol{p}_i), \tag{6.9}$$

where now the number of basis functions N can easily be much smaller than the number of pixels M.

For a given data set, some types of basis function will provide more natural and sparse representation than others. We can further generalise (6.9) using parameterised dictionary learning, by fitting different families of standard basis functions (determined by a categorical variable T). The optimisation then determines T, as well as each basis function's amplitude a_i and parameters $\boldsymbol{p}_i$ by reconstructing the signal as

$$f(\boldsymbol{x}; T, \boldsymbol{a}, \boldsymbol{p}_1, \ldots, \boldsymbol{p}_N) = \sum_{i=1}^{N} a_i \phi^{(T)}(\boldsymbol{x}; \boldsymbol{p}_i). \tag{6.10}$$

Commonly used basis function families include Gaussians, wavelets and shapelets.

6.3 A Bayesian approach

6.3.1 Bayesian formulation of regression and regularisation

Before introducing our full Bayesian sparse reconstruction framework in Section 6.3.3, we first give a Bayesian formulation of the regression and regularisation problems discussed

in Sections 6.2.1 and 6.2.2 as the comparison is very informative. In these cases the number, type and shape of basis functions are fixed and the only parameters of the model are the amplitudes — i.e. $\boldsymbol{\theta} = \boldsymbol{a}$.

In general, defining the likelihood of the basis function fit given some data $\mathcal{D}$ requires knowledge of how measurement errors are distributed. For example, a common assumption in the literature is that there are independent Gaussian errors on the signal values $\{y_d\}$, and no errors on the data points' coordinates $\{\boldsymbol{x}_d\}$. In this case the likelihood of the data given the model is

$$\begin{aligned} \mathcal{L}(\boldsymbol{a}) = P(\mathcal{D}|\boldsymbol{a}, \mathcal{M}) &= \prod_{d=1}^{D} \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(y_d - f(\boldsymbol{x}_d; \boldsymbol{a}))^2}{2\sigma_y^2}\right) \\ &\propto \exp\left(-\frac{\|\boldsymbol{y} - \Phi\boldsymbol{a}\|_2^2}{2\sigma_y^2}\right), \end{aligned} \tag{6.11}$$

recovering the (exponentiated) least squares objective function from (6.3). Of course this assumption may be inappropriate for some data sets. For example, for low-level photon-counting measurements a Poisson likelihood function (or similar) may be required. Although the Bayesian formulation can naturally accommodate other likelihood functions, for simplicity we will henceforth consider only independent Gaussian errors.

In order for the Bayesian approach to give the same maximum *a posteriori* parameter values as the optimisation in Section 6.2.2, the prior $\pi(\boldsymbol{a})$ must correspond to the exponential of the regularisation terms in (6.4-6.7). For a more formal derivation of this result in the context of the Wiener filter, see Hobson et al. (1998) and Lasenby et al. (2001).

In the Bayesian framework, the different regularisation techniques in (6.4-6.7) are analogous to the following choices of priors:

- L_2 (squared) regularisation (6.4) corresponds to a Gaussian prior:

$$\pi(\boldsymbol{a}) \propto \exp(-\lambda\|\boldsymbol{a}\|_2^2). \tag{6.12}$$

- L_1 regularisation (6.5) corresponds to a Laplacian prior:

$$\pi(\boldsymbol{a}) \propto \exp(-\lambda\|\boldsymbol{a}\|_1). \tag{6.13}$$

- L_0 regularisation (6.6) corresponds to an exponential prior on the number of non-zero components of $\boldsymbol{a}$:

$$\pi(\boldsymbol{a}) \propto \exp(-\lambda \|\boldsymbol{a}\|_0) = \exp(-\lambda N), \tag{6.14}$$

  where N is the number of basis functions used in the signal reconstruction.

- Entropy regularisation (6.7) corresponds to an entropic prior

$$\pi(\boldsymbol{a}) \propto \exp(\lambda S(\boldsymbol{a})), \tag{6.15}$$

  where $S(\boldsymbol{a})$ is defined in (6.7).

More generally other priors can be used. For example, one may promote sparsity by using any prior which has fatter tails than a Gaussian and is also more concentrated at zero — such priors prefer to shrink amplitudes to zero while also being lenient in allowing larger amplitudes. Thus, as an alternative to the Laplacian distribution (6.13), one could use for example a Cauchy distribution

$$\pi(\boldsymbol{a}) = \prod_{j=1}^{M} \frac{\lambda}{\pi} \frac{1}{\lambda^2 + a_j^2}. \tag{6.16}$$

In addition, the L_2 regularisation prior (6.12) can be generalised to include some covariance matrix $\boldsymbol{C}$, which may be a function of some further parameters $\boldsymbol{\theta}$,

$$\pi(\boldsymbol{a}) \propto \exp(-\lambda \boldsymbol{a}^\intercal \boldsymbol{C}^{-1} \boldsymbol{a}). \tag{6.17}$$

This form can be used to reconstruct a signal as a Gaussian process (see Rasmussen, 2004, for an introduction), with $\boldsymbol{C}$ representing its correlation structure. When performing the optimisation, the basis matrix Φ most naturally contains Fourier modes. The optimisation is typically performed by selecting both $\boldsymbol{\theta}$ and λ to maximise the Bayesian evidence (sometimes the value of λ is chosen *a priori*). Indeed, Gaussian processes could be further generalised by using a different form for the prior term — for example "entropic processes" with $\pi(\boldsymbol{a}) \propto \mathrm{e}^{-\lambda S(\boldsymbol{La})}$, where S is defined as in (6.7) and $\boldsymbol{C} = \boldsymbol{LL}^\intercal$ is the Cholesky decomposition of the signal correlation matrix (Hobson et al., 1998).

6.3.2 Sampling and model selection

Maximum *a posteriori* estimates of the parameters $\boldsymbol{a}$ can be found from the posterior distribution $\propto \mathcal{L}(\boldsymbol{a})\pi(\boldsymbol{a})$ in an analogous manner to the optimizations in (6.3-6.7). However, a major advantage of the Bayesian approach is that it provides a *generative* model and allows the full posterior distribution to be sampled. This provides additional information such as posterior distributions on the weights $\boldsymbol{a}$ and other quantities of interest.

Furthermore, the posterior distribution allows the appropriate number of basis functions to be chosen via Bayesian model selection by calculating posterior odds ratios (2.15). This naturally penalises more complex models and, with an appropriate choice of priors, provides a principled Bayesian method for creating models with the level of complexity which is justified by the data. Finally one can either choose the maximum *a posteriori* number of basis functions or, better, marginalise over N so the fit with each number of basis functions is weighted in proportion to its posterior probability. Any *a priori* expectation of the degree of sparsity can be included in the priors, and there is no need for an additional regularization term.

6.3.3 Bayesian sparse reconstruction

Following the discussion in the previous sections, we propose reconstructing the relationship $y = f(\boldsymbol{x})$ as a sum of N basis functions $\phi^{(T)}$ of type T with weights a_i and shape and location parameters $\boldsymbol{p}_i$ as

$$f(\boldsymbol{x}; T, N, \boldsymbol{a}, \boldsymbol{p}_1, \ldots, \boldsymbol{p}_N) = \sum_{i=1}^{N} a_i \phi^{(T)}(\boldsymbol{x}, \boldsymbol{p}_i). \tag{6.18}$$

One can then perform Bayesian inference over the full parameter space of $\boldsymbol{\theta} = (T, N, \boldsymbol{a}, \boldsymbol{p}_1, \ldots, \boldsymbol{p}_N)$.

This approach has the desirable properties that:

- full posterior distributions on parameters can be recovered by sampling (rather than simply optimizing);
- sparsity can be enforced directly through priors on the total number of basis functions N;
- there are a variable number of basis functions with variable positions;

- there is no need to choose a regularisation constant λ;
- families and/or shapes of basis functions are determined and can be marginalised over;
- arbitrary constraints can be imposed on the reconstruction (not just positivity);
- any type of noise can be included — e.g. Gaussian, Poisson, etc. If the size or nature of the noise is unknown, it can be expressed in terms of additional parameters which can be marginalised over;
- missing and/or irregular data can be accommodated;
- the model is generative and can easily be extended to deconvolution.

The remainder of this section discusses how Bayesian sparse reconstructions can be computed, with numerical tests presented in the following section.

6.3.4 Vanilla and adaptive methods

Given some noisy signal to be reconstructed, Bayesian model selection can be used to determine an appropriate type T and number N of basis functions to use by calculating the Bayesian evidence $\mathcal{Z}_{T,N}$ for the fit using each combination (model) T, N. Using (2.14), the posterior probability of each model is proportional to $\mathcal{Z}_{T,N}\Pi_{T,N}$, where $\Pi_{T,N}$ is the prior probability of the model and over-complex models are penalised by lower evidences. We term this the *vanilla method*. One can then either select the model with the highest posterior probability or, better, use a combination of all models weighted by their posterior probability ("multi-model analysis").

The *adaptive method* (mentioned in Section 3.7) is an alternative product-space approach, which analyses a "meta-model" containing one or more discrete parameters with values corresponding to each individual model. The likelihood of a sample is found by selecting the model indicated by the discrete parameters, then working out the likelihood for this model using the remaining parameters. A fixed dimensionality which is sufficient for the individual model with the most parameters is used; for models with fewer parameters, the likelihood is independent of the remaining unneeded parameters. This is an alternative to transdimensional sampling methods such as reversible-jump

MCMC (Green, 1995). Hee et al. (2016, 2017) used the adaptive method in reconstructing 1-dimensional signals by linearly interpolating between N points ("nodes"), with their co-ordinates as free parameters. We generalise this approach by letting the integer parameter N represent the number of basis functions (of any dimension) to be used, and when needed also including a second integer parameter T to determine the form of the basis functions. Posterior distributions of T and N are found using parameter estimation.

6.3.5 Practical considerations for sampling the posterior

The posterior is typically of moderate to large dimensionality, and will be non-convex and multimodal with pronounced degeneracies. Furthermore, due to the integer parameters T and N, methods requiring gradients cannot be used. We explore the posterior using nested sampling (Skilling, 2006), which is well suited to such problems and can be performed using software packages such as `MultiNest` (Feroz and Hobson, 2008; Feroz et al., 2008, 2013) or `PolyChord` (Handley et al., 2015a,b). The adaptive method calculates posterior odds ratios indirectly via parameter estimation by sampling the integer parameters T and N, and as a result its sampling errors have the characteristics described in Chapter 3. In contrast the vanilla method uses direct evidence calculations.

Dynamic nested sampling (discussed in Chapter 5) gives large efficiency gains for parameter estimation, meaning it works well with the adaptive method. In contrast the efficiency gains for evidence calculations are relatively modest (except in low dimensions), so dynamic nested sampling only produces small speedups for calculations of posterior odds with the vanilla method and we do not use it in this case. Results in this chapter were calculated using `dyPolyChord` (Higson, 2018a) — a dynamic nested sampling package based on `PolyChord`. Due to the challenging multimodal posteriors produced by the integer parameter in the adaptive method, we use a large fraction (50%) of the total computational budget for each calculation on `dyPolyChord`'s initial exploratory run. This reduces the possible efficiency gain, but `dyPolyChord` is still able to produce significant speedups compared to standard nested sampling.

6.4 Fitting 1-dimensional data

We first demonstrate Bayesian sparse reconstruction by finding the dependence of some scalar quantity y on another scalar variable x, and to make the example more challenging we allow errors on both the data values y_d and positions x_d.

If each measurement has an independent error distribution $P(x_d, y_d|X_d, Y_d)$ about its true value X_d, Y_d then the probability of the observed data given some set of true values is

$$P(\mathcal{D}|\{X_d, Y_d\}) = \prod_{d=1}^{D} P(x_d, y_d|X_d, Y_d). \tag{6.19}$$

The unknown true data values X_d, Y_d are then marginalised out using the basis fitting model by taking $Y_d = f(X_d; T, N, \boldsymbol{a}, \boldsymbol{p}_1, \ldots, \boldsymbol{p}_N)$ and integrating over the distribution of the x coordinates at which data points were sampled $P(X_d)$. Hence each likelihood call involves an integral for every data point:

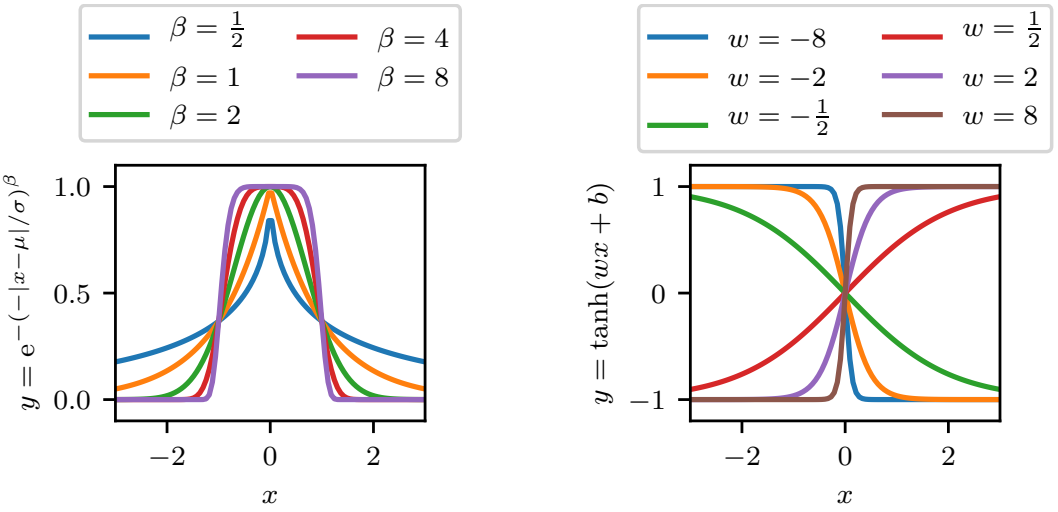
$$P(\mathcal{D}|P(X_d), T, N, \boldsymbol{a}, \boldsymbol{p}_1, \ldots, \boldsymbol{p}_N) \quad = \quad \prod_{d=1}^{D} \int P(x_d, y_d|X_d, f(X_d))\, P(X_d)\, \mathrm{d}X_d, \tag{6.20}$$

where for brevity we have omitted the dependence of $f(X_d; T, N, \boldsymbol{a}, \boldsymbol{p}_1, \ldots, \boldsymbol{p}_N)$ on the parameters $T, N, \boldsymbol{a}, \boldsymbol{p}_1, \ldots, \boldsymbol{p}_N$.

We first consider samples to be taken uniformly in the range $X_- < X_d < X_+$ with independent Gaussian x and y errors of size σ_x and σ_y. In this case (6.20) gives the likelihood (Hee et al., 2016)

$$\begin{aligned}\mathcal{L}(T, N, \boldsymbol{a}, \boldsymbol{p}_1, \ldots, \boldsymbol{p}_N) =& P(\mathcal{D}|T, N, \boldsymbol{a}, \boldsymbol{p}_1, \ldots, \boldsymbol{p}_N) \\ =& \prod_{d=1}^{D} \int_{X_-}^{X_+} \frac{\exp\left[-\frac{(x_d - X_d)^2}{2\sigma_x^2} - \frac{(y_d - f(X_d))^2}{2\sigma_y^2}\right]}{2\pi\sigma_x\sigma_y(X_+ - X_-)}\, \mathrm{d}X_d.\end{aligned} \tag{6.21}$$

The priors on the parameters and models can be specified as required, and together with the likelihood can be used to sample numerically from the posterior and calculate evidences. As $f(x; T, N, \boldsymbol{a}, \boldsymbol{p}_1, \ldots, \boldsymbol{p}_N)$ is typically invariant under interchange of basis function index the prior space can be shrunk by a factor of $N!$ by enforcing ordering using "forced identifiability" (sorted) priors; see Handley et al. (2015b, Appendix A2) for a more detailed discussion.



(a) Generalised Gaussians (6.22) for different values of β; each has $\mu = 0$ and $\sigma = 1$.

(b) tanh functions (6.23) with different values of w. All the lines shown use $b = 0$.

Figure 6.5: Illustrations of 1-dimensional basis functions.

6.4.1 Basis functions

The likelihood (6.21) applies for mixture models with any 1-dimensional basis function; we demonstrate it using 1-dimensional generalised Gaussians

$$\phi^{(\mathrm{g1d})}(x; \boldsymbol{p}) = \phi^{(\mathrm{g1d})}(x, \mu, \sigma, \beta) = \mathrm{e}^{-(|x-\mu|/\sigma)^\beta} \tag{6.22}$$

and 1-dimensional tanh functions

$$\phi^{(\mathrm{t1d})}(x, \boldsymbol{p}) = \phi^{(\mathrm{t1d})}(x, w, b) = \tanh(wx + b). \tag{6.23}$$

Their shape and location are determined by parameters $\boldsymbol{p} = (\mu, \sigma, \beta)$ and $\boldsymbol{p} = (w, b)$ respectively; the effects of different parameters are illustrated in Figure 6.5. The magnitude of each basis function in the fit is controlled by an amplitude parameter a.

When $\beta = 2$, (6.22) is proportional to a normal distribution with variance $\sigma^2/2$, and when $\beta = 1$ it is proportional to a Laplace distribution. For large values of β, (6.22) is approximately uniform $\in [\mu - \sigma, \mu + \sigma]$ and zero elsewhere. The normalisation constant $\beta/(\Gamma(\frac{1}{\beta})2\sigma)$ is omitted from (6.22) as it causes pronounced degeneracies in the joint posterior distributions of a, β and σ due to all 3 parameters affecting the height of the basis function at its centre.

The priors used for the basis functions are shown in Table 6.1. The exponential prior on the amplitudes $\boldsymbol{a}$ of the generalised Gaussians has the desirable property that

Parameter	Prior Type	Prior Parameters
1-dimensional generalised Gaussian (6.22)		
N	Uniform (integer)	$\in \mathbb{Z} \cap [1, 5]$
a	Sorted Exponential	$\lambda = 1$
μ	Uniform	$\in [0, 1]$
σ	Uniform	$\in [0.03, 1.0]$
β	Exponential	$\lambda = 0.5$
1-dimensional tanh (6.23)		
N	Uniform (integer)	$\in \mathbb{Z} \cap [1, 5]$
a	Sorted Half Gaussian	$\mu = 0, \sigma = 5$
w	Gaussian	$\mu = 0, \sigma = 5$
b	Gaussian	$\mu = 0, \sigma = 5$
Adaptive basis function family selection		
T	Uniform (integer)	$\in \mathbb{Z} \cap [1, 2]$
2-dimensional generalised Gaussian (6.25)		
N	Uniform (integer)	$\in \mathbb{Z} \cap [1, 5]$
a	Sorted Exponential	$\lambda = 1$
μ_1	Uniform	$\in [0, 1]$
μ_2	Uniform	$\in [0, 1]$
σ_1	Uniform	$\in [0.03, 0.5]$
σ_2	Uniform	$\in [0.03, 0.5]$
β_1	Exponential	$\lambda = 0.5$
β_2	Exponential	$\lambda = 0.5$
Ω	Uniform	$\in [-\pi/4, \pi/4]$

Table 6.1: Priors on basis function parameters used in this chapter. Sorted priors have ordering enforced; see Handley et al. (2015b, Appendix A2) for more details. The half Gaussian prior on the amplitudes of the tanh basis functions is truncated at zero and permits only positive values.

it is a function of only the sum of the amplitudes and does not vary based on how the total is split between basis functions. We use a uniform prior on the generalised Gaussians' σ, rather than a scale prior favouring smaller values, as we find the latter causes overfitting by encouraging the addition of narrow generalised Gaussians to fit noise in the data. The priors on the tanh basis functions are chosen for consistency with the neural networks discussed in Chapter 7.

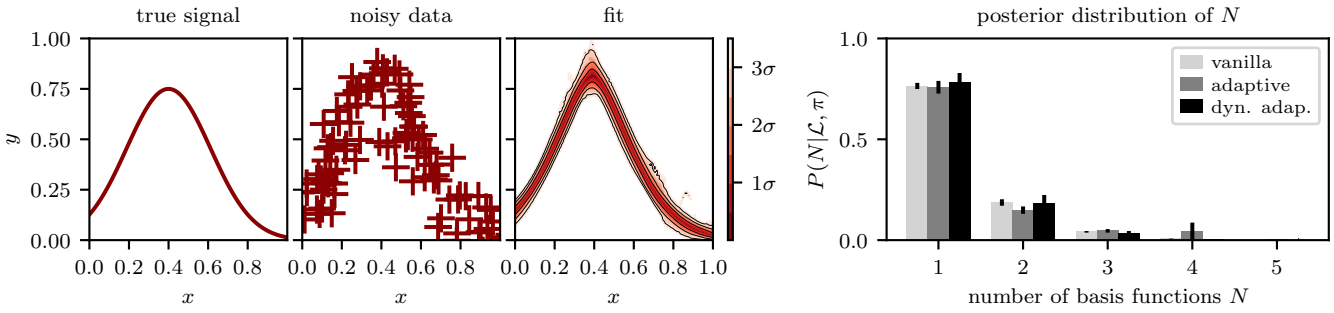
6.4.2 Numerical results

We first illustrate Bayesian sparse reconstruction using simulated 1-dimensional data points sampled from basis function mixture models. Independent Gaussian x- and y-errors of size $\sigma_x = \sigma_y = 0.07$ are added to each data point.

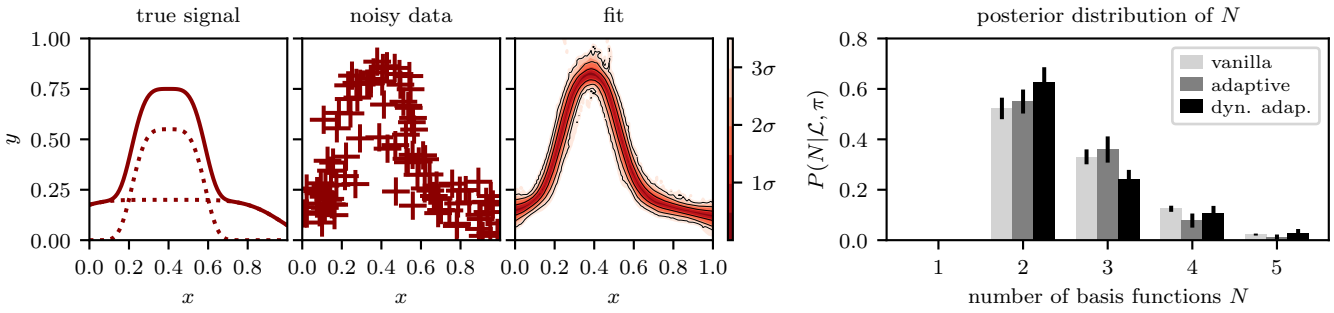
Figure 6.6 show the results from fitting different Gaussian mixture models; plots of the posterior distribution of y were created using the `fgivenx` package (Handley, 2018). Despite the large measurement noise and visually similar data in Figures 6.6a to 6.6c, our approach is able to correctly reconstruct the different true signals and identify the increasing number of basis functions required to model each signal.

Figure 6.7 shows examples of signal reconstructions conditioned on specific numbers of basis functions, and illustrates the effect of increasing N on the fit produced. Such plots can be calculated from adaptive method nested sampling runs by marginalising over different values for the integer parameter N. However, we use the vanilla method runs to make these plots as the adaptive method dedicates relatively few samples to exploring the highly disfavoured N values with make negligible contribution to the overall fit. This is a desirable feature which makes fitting with the adaptive method more efficient, but as a consequence the vanilla method can produce more accurate plots conditioned on disfavoured values of N.

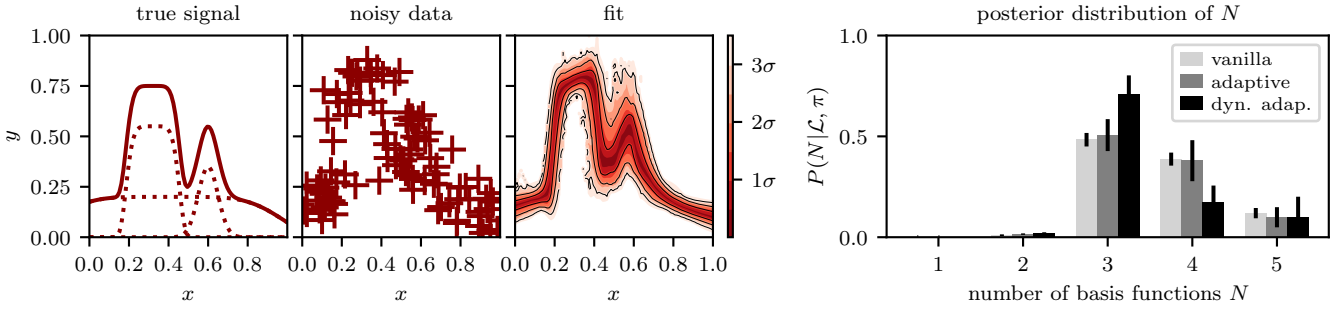
Figures 6.8 and 6.9 show examples of fitting tanh basis functions to 1-dimensional data, and are similar to Figures 6.6 and 6.7. As for the generalised Gaussian basis functions, our approach is able to accurately reconstruct the true signal from the noisy data and identify the increasing complexity of the successive signals in Figures 6.8a to 6.8c. However it is not necessarily the case that the most probable *a posteriori* value of N, given the noisy data and priors, is the same as the number of basis functions from which the data was sampled; in Figure 6.8c, $P(N = 4|\mathcal{L}, \pi)$ and $P(N = 5|\mathcal{L}, \pi)$ are greater than $P(N = 3|\mathcal{L}, \pi)$.



(a) Data from a single generalised Gaussian.



(b) Data from the sum of two generalised Gaussians.



(c) Data from the sum of three generalised Gaussians.

Figure 6.6: Fitting generalised Gaussian basis functions to 100 data points sampled from different combinations of basis functions. In each row the first plot shows the true signal (a sum of basis functions); where this contains more than one basis function, the individual components are shown with dashed lines. The data, which includes added normally distributed x- and y-errors with $\sigma_x = \sigma_y = 0.07$, is show in the second plot. The third plot shows the fit calculated using the adaptive method with dynamic nested sampling; coloured contours represent posterior iso-probability credible intervals on $y(x)$. The bar plots on the right display the posterior distribution for different numbers of basis functions N; values calculated using the vanilla method and using the adaptive method with standard nested sampling are also included for comparison. Results shown for the adaptive method use a combined inference from 5 runs, each of which computes a full posterior on N and uses 1,000 live points; adaptive runs using dynamic nested sampling have `dyPolyChord` settings $n_{\text{init}} = 500$ and `dynamic_goal` $= 1$. Results for the vanilla method use 5 separate runs, each with 200 live points, to compute the evidence for each value of N. All runs use the setting `num_repeats` $= 100$. The parameters of the basis functions in the true signal and numerical results for the computational efficiency of the different methods are shown in Table 6.3 and Table 6.6 respectively in Appendix 6.C.

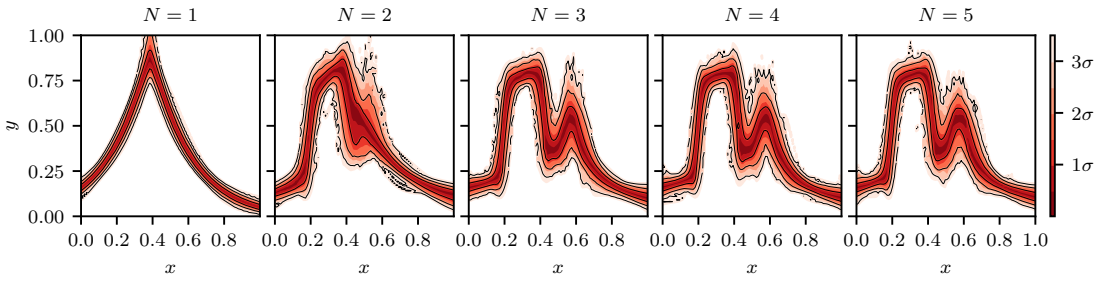


Figure 6.7: Fits of the data shown in Figure 6.6c conditioned on different numbers N of basis functions. These plots are made using the vanilla method nested sampling runs, as the adaptive method runs contain relatively few samples from the heavily disfavoured values of N.

6.4.3 Adaptive basis function families

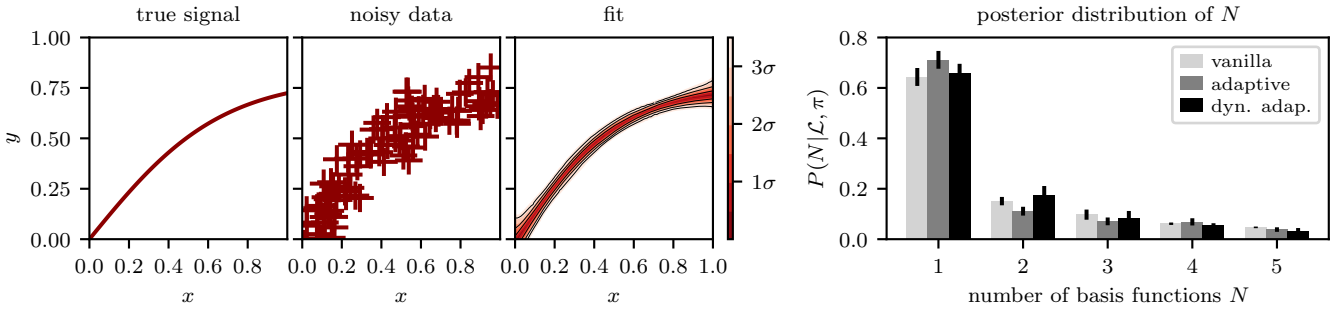
We now illustrate including a second integer parameter T which selects the basis function family, in addition to N which selects the number of basis functions given the family. Figure 6.10 shows fits using both generalised Gaussians ($T = 1$) and tanh functions ($T = 2$), with a uniform prior on $T \in \mathbb{Z} \cap [1, 2]$.

The generalised Gaussians are a much better fit for the data in Figure 6.10a, with posterior probability from the adaptive method with dynamic nested sampling of $P(T = 2|\mathcal{L}, \pi) = (1 \pm 1) \times 10^{-7}$. In contrast the two families are competitive for the data in Figure 6.10b, with $P(T = 2|\mathcal{L}, \pi) = 0.6 \pm 0.1$ indicating only a weak favouring of the tanh basis function. These results are, however, highly dependent on the priors used for the basis functions' parameters.

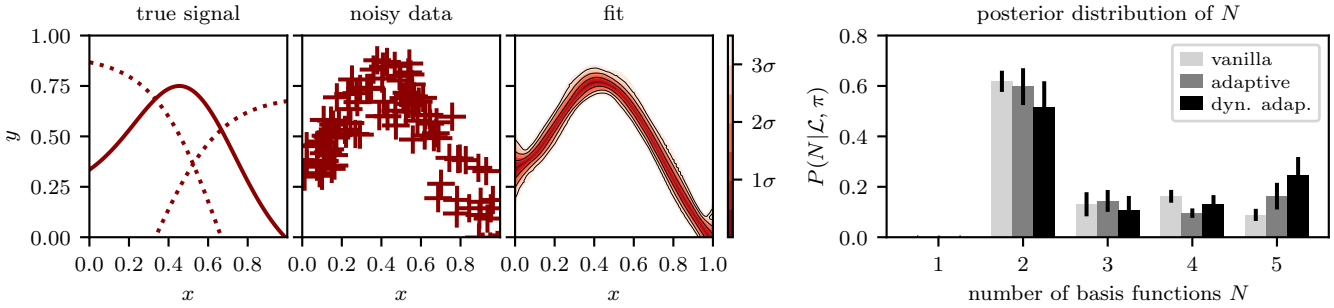
A possible application of this adaptive selection of basis function families T would be to compare different parametric models for sources in astronomical images in which the true number of sources is unknown. In this case computing a posterior distribution on T would not only marginalise over the distributions of the sources' parameters, but also over the unknown number of sources N.

6.4.4 Comparison of vanilla and adaptive results

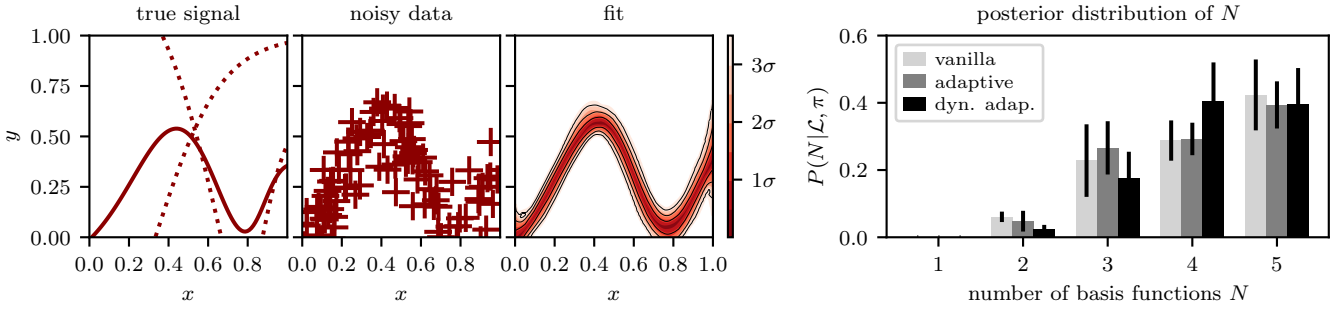
The adaptive method allows significant improvements in accuracy of the overall fit for a given computational cost by allocating fewer samples into disfavoured models which make a small or negligible contribution to the output. In addition, by transforming the model selection from evidence calculations (as in the vanilla method) to a parameter



(a) Data from a single tanh basis function.



(b) Data from the sum of two tanh basis functions.



(c) Data from the sum of three tanh basis functions.

Figure 6.8: As for Figure 6.6 but using tanh basis functions instead of generalised Gaussians. The parameters of the tanh basis functions in true signal are shown in Table 6.4 in Appendix 6.C.

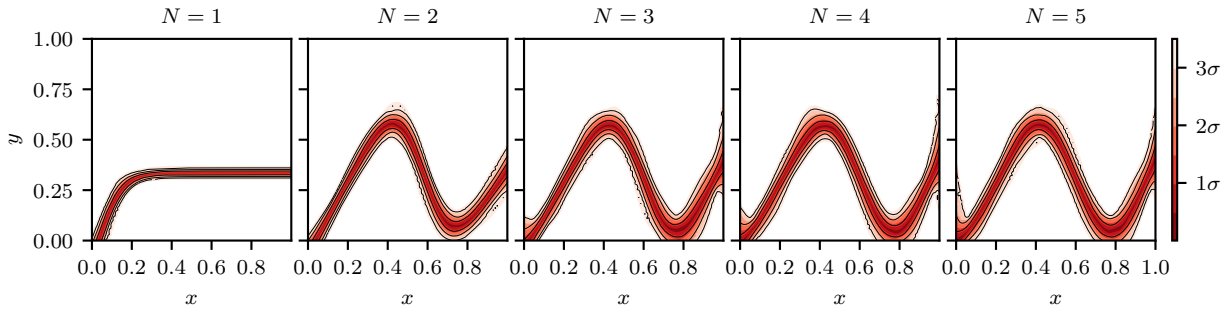
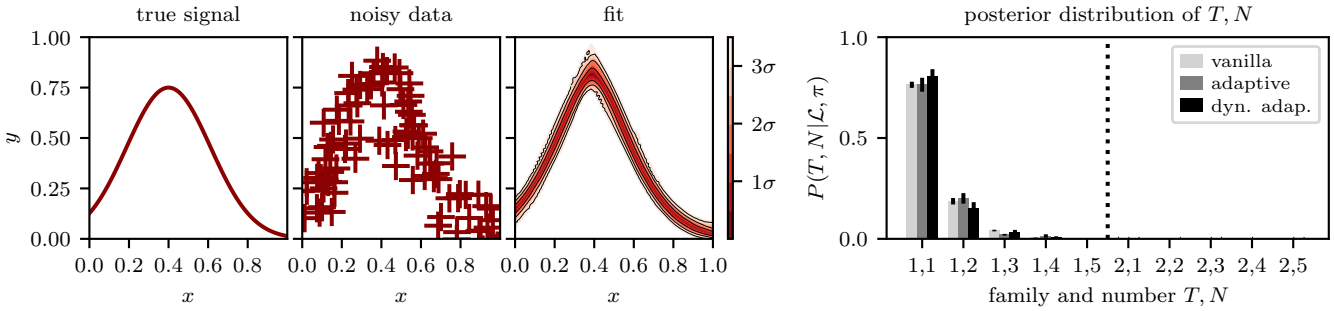
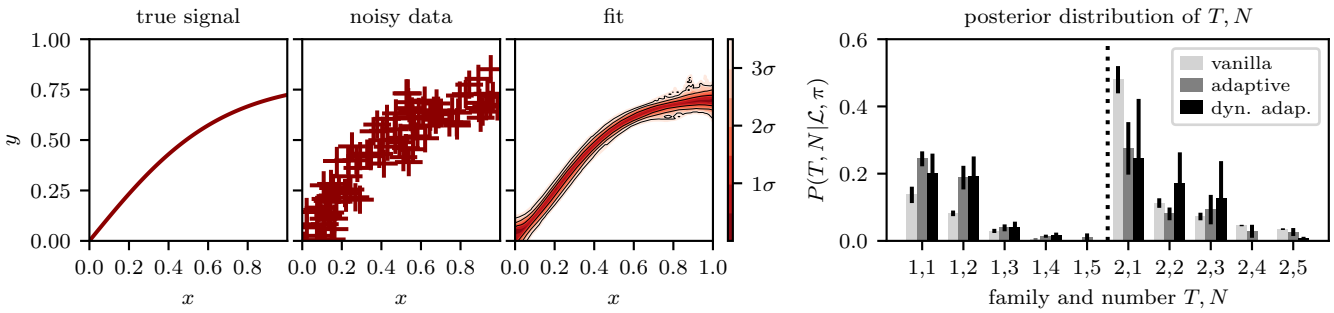


Figure 6.9: Fits of the data sets shown in Figure 6.8c conditioned on different numbers N of basis functions.



(a) Data used in Figure 6.6a from a single generalised Gaussian.



(b) Data used in Figure 6.8a from a single tanh function.

Figure 6.10: Fitting generalised Gaussian and tanh basis functions to data in a fully adaptive manner with the family determined by an integer parameter T. In each row the first plot shows the true signal (a sum of basis functions). The data, shown in the second plot, contains normally distributed x- and y-errors with $\sigma_x = \sigma_y = 0.07$. The third plot shows the fit calculated using the adaptive method with dynamic nested sampling; coloured contours represent posterior iso-probability credible intervals on $y(x)$. The bar plots on the right display the posterior posterior distribution on different families T and numbers of basis functions N; values calculated using the vanilla method and using the adaptive method with standard nested sampling are also included for comparison. Results for the adaptive method use a combined inference from 5 runs, each of which computes a full posterior on T, N and uses 1,000 live points; adaptive runs using dynamic nested sampling have `dyPolyChord` settings $n_{\text{init}} = 500$ and `dynamic_goal` $= 1$. Results for the vanilla method use separate runs, each with 200 live points, to compute the evidence for each combination of T and N. All runs use the setting `num_repeats` $= 100$.

estimation problem on N, the adaptive method changes the nature of the sampling errors. Uncertainty in the rate of shrinkage at each step before any significant posterior mass is reached — the dominant source of error in nested sampling evidence calculations — has a negligible effect on parameter estimation of the posterior distribution of N. This can allow order-of-magnitude gains in computational efficiency of posterior odds ratios from the adaptive method compared to the vanilla method, as observed by Chua et al. (2018). However, a downside of the method is that including all the models and the integer parameter makes the posterior distribution highly multimodal and more challenging for the sampler to explore.

Following (5.7), we measure the computational efficiency gains from alternative methods compared to the vanilla method with standard nested sampling as

$$\text{efficiency gain} = \frac{\text{Var}[\text{vanilla NS results}]}{\text{Var}[\text{method NS results}]} \times \frac{\overline{N_{\text{samp,van}}}}{\overline{N_{\text{samp,meth}}}}. \tag{6.24}$$

Here the first term is the ratio of the estimated variance of the results of repeated calculations using the vanilla method and the alternative method; the second term is the ratio of the mean number of samples from the nested sampling runs using each method. Numerical results for the efficiency gains from the different methods are show in Table 6.6 in Appendix 6.C. These use estimates of the variance of results calculated using the bootstrap resampling method described in Chapter 3, which avoids the need to compute large numbers of nested sampling runs but also does not include additional errors due to implementation-specific effects. As described in Appendix 6.C, we find that the sampler is not able to explore the parameter space perfectly with the settings used, meaning the true variance of results is higher than the bootstrap estimates. As a result, given the adaptive method's more complex posterior distribution, the efficiency gains of factors of up to 14 ± 3 for the adaptive method and 46 ± 9 for the adaptive method using dynamic nested sampling are likely to be overestimates. These efficiency gains are best viewed as an indication of what is possible using the method with more computational power — such as using a higher value for `dyPolyChord`'s `num_repeats` setting.

6.5 2-dimensional image fitting

We now demonstrate Bayesian sparse reconstruction for monochrome images. Here, for each data point (pixel) d, $\boldsymbol{x}_d = (x_1, x_2)_d$ is the pixel location and $y_d \in [0, 1]$ is the scalar signal. For simplicity we assume that the errors in the pixel positions $\boldsymbol{x}_d$ are negligible, and consider the case that the signal y_d for each pixel contains independent Gaussian noise with size $\sigma_y = 0.2$ — in this case the likelihood is given by (6.11).

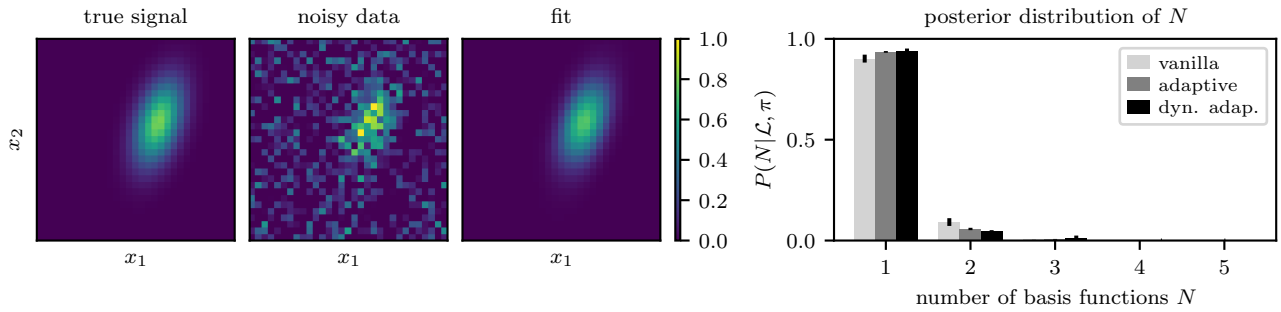
We define 2-dimensional generalised Gaussians as the product of two 1-dimensional generalised Gaussians (6.22) rotated by angle Ω around their mean $\boldsymbol{\mu}$:

$$\begin{aligned} \phi^{(\mathrm{g2d})}(\boldsymbol{a}, \boldsymbol{p}) &= \phi^{(\mathrm{g2d})}(\boldsymbol{x}, \boldsymbol{\mu}, \boldsymbol{\sigma}, \boldsymbol{\beta}, \Omega) \\ &= \phi^{(\mathrm{g1d})}(x_1', \mu_1, \sigma_1, \beta_1) \times \phi^{(\mathrm{g1d})}(x_2', \mu_2, \sigma_2, \beta_2) \\ \text{where} \quad \boldsymbol{x'} &= \boldsymbol{\mu} + (\boldsymbol{x} - \boldsymbol{\mu}) \begin{pmatrix} \cos(\Omega) & -\sin(\Omega) \\ \sin(\Omega) & \cos(\Omega) \end{pmatrix}. \end{aligned} \tag{6.25}$$

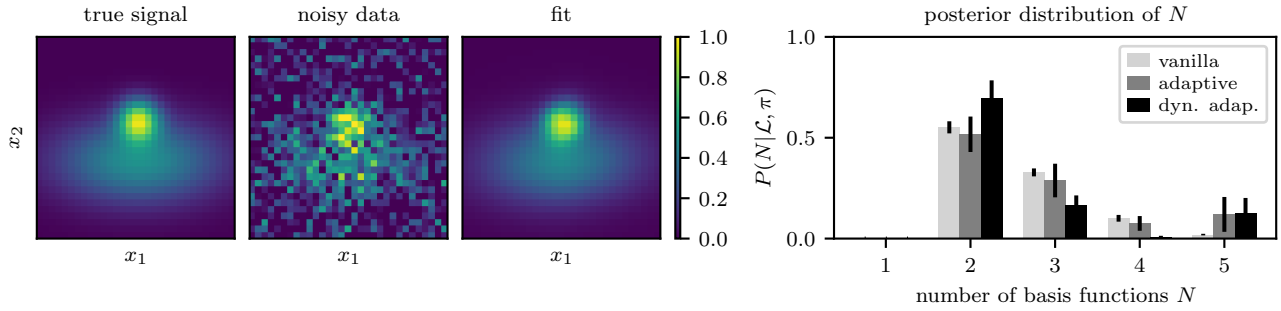
The priors used are shown in Table 6.1.

Figure 6.11 show examples of Bayesian sparse reconstruction fitting 2-dimensional images. The fits show the mean values predicted for each pixel, averaged over all the samples produced in proportion to their posterior weight. Using the mean value avoids overfitting — which would occur if, for example, the fit was simply calculated from the sample with the highest likelihood (the maximum likelihood estimate). The samples provide a full posterior distribution on the parameters and output signal, so other quantities such as the uncertainty on each pixel can also be easily calculated. Figure 6.12 shows fits conditioned on specific values of N, and illustrates how increasing the number of basis functions allows increasingly complex structure to be included in the recovered image.

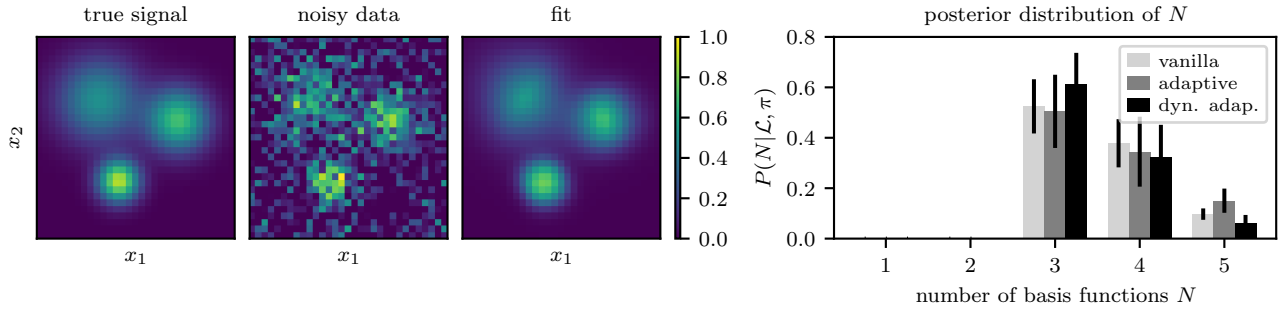
As in the 1-dimensional case, our approach is able to faithfully reconstruct the signal from the noisy data and the numbers of basis functions with the highest posterior probability (shown in the bar charts on the right of each subfigure) match the number of components in the mixture model used for the signal. Furthermore, Table 6.7 in Appendix 6.C shows efficiency gains (6.24) from the adaptive method of up to 10 ± 2 and from the adaptive method with dynamic nested sampling of up to 16 ± 3 in these cases. However, as discussed in Section 6.4.4, these numbers may overestimate the efficiency gains observed in practice with the settings used.



(a) Image of a single 2-dimensional generalised Gaussian.



(b) Image of the sum of two 2-dimensional generalised Gaussians.



(c) Image of the sum of three 2-dimensional generalised Gaussians.

Figure 6.11: Fitting 2-dimensional generalised Gaussian basis functions to 32×32 images of mixtures of generalised Gaussians. In each row the 2 plots on the left show the true signal and the data, which includes added normally distributed y-errors with $\sigma_y = 0.2$. The third column shows the mean value of $y(\boldsymbol{x})$ from the posterior samples produced using the adaptive method with dynamic nested sampling. The bar plots on the right display the posterior distribution for different numbers of basis functions N; values calculated using the vanilla method and adaptive method without dynamic nested sampling are also included for comparison. Results for the adaptive method show a combined inference from 5 runs, each of which computes a full posterior on N and uses 2,000 live points; adaptive runs using dynamic nested sampling have `dyPolyChord` settings $n_{\text{init}} = 1,000$ and `dynamic_goal` $= 1$. Results for the vanilla method use 5 separate runs, each with 400 live points, to compute the evidence for each value of N. All runs use the setting `num_repeats` $= 250$. The parameters of the basis functions in the true signal and numerical results for the computational efficiency of the different methods are shown in Table 6.3 and Table 6.6 respectively in Appendix 6.C.

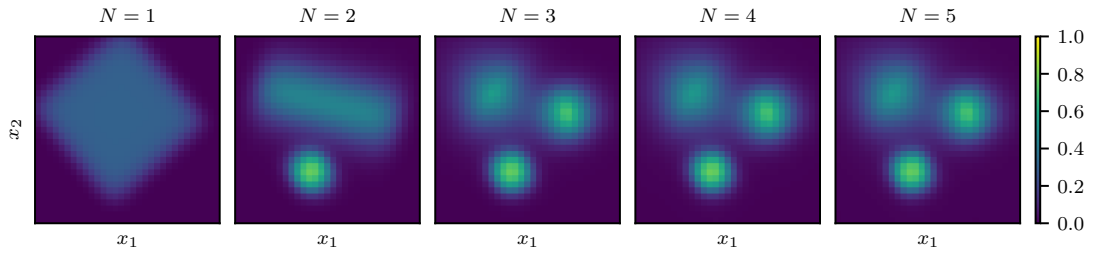


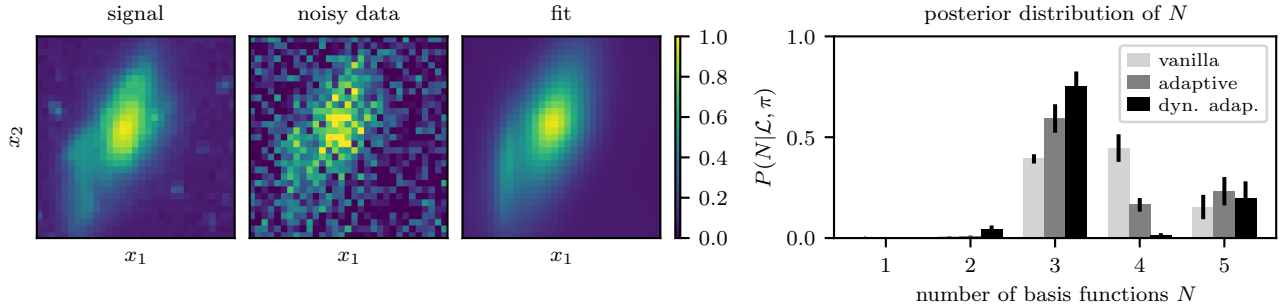
Figure 6.12: Fits of the data in Figure 6.11c conditioned on different numbers N of basis functions; these plots use results from the vanilla method.

6.5.1 Application to astronomical images

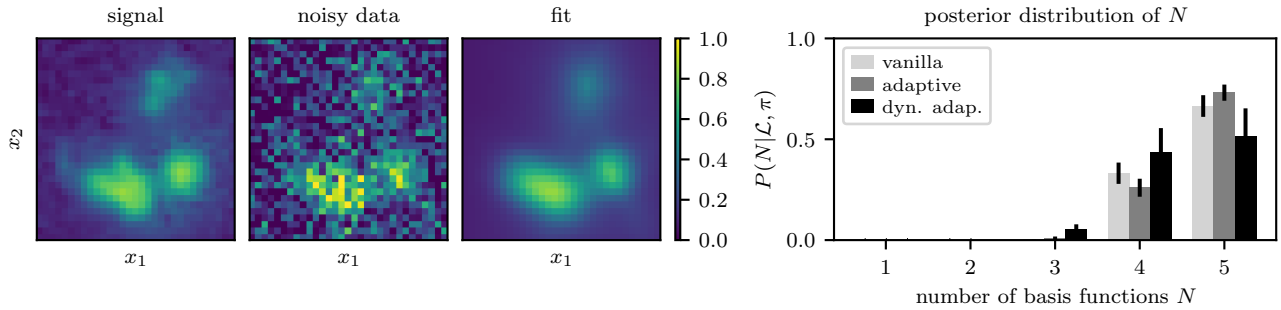
We now apply the 2-dimensional fitting techniques from the previous section to astronomical images from the Hubble Space Telescope eXtreme Deep Field (Illingworth et al., 2013). These are not "true signals" as in the previous examples because the images contain some measurement uncertainty, but this is relatively small compared to our added Gaussian errors of $\sigma_y = 0.2$. We therefore use them as an approximation of a realistic physical signal for testing our method. Furthermore, for this first trial application of our method, we provide only a visual demonstration of the accuracy of our image reconstructions (to be assessed qualitatively). A more quantitative evaluation can be performed in the future using simulations where the noise-free signal values are available.

Figure 6.13 shows fitting images of galaxies from the Hubble deep field using 2-dimensional generalised Gaussians (6.25), and Figure 6.14 shows fits of specific numbers of basis functions (marginalised for different values of N). Our method is able to faithfully reconstruct the signal from the noisy data, as can be seen from a visual comparison of the fit and the signal. In this case, with the settings used, the posterior distributions of N show some inconsistencies between the different methods. These occur as in order to explore the challenging posterior consistently, `PolyChord` and `dyPolyChord` require higher live points and/or `num_repeats` setting than those used; this leads to additional random errors. However this lack of precision in the posterior probabilities of N has little negative impact on the overall fit, as in each case all the posterior mass is allocated to values of N which provide good representations of the data.

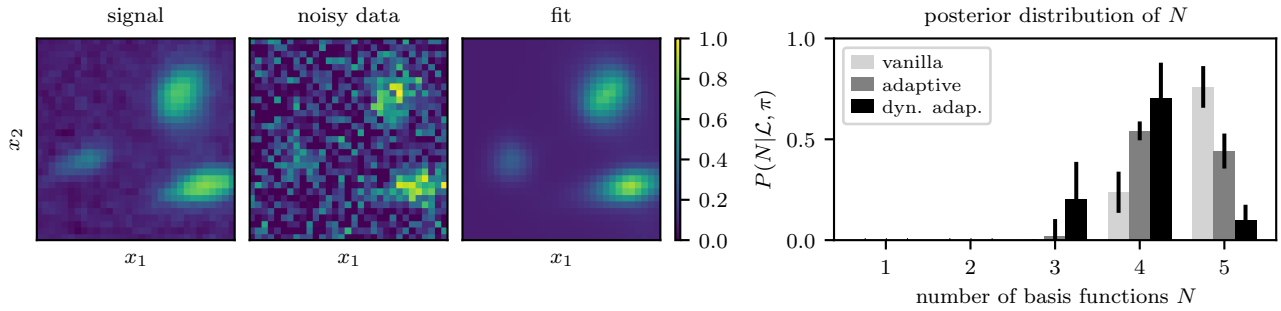
The posterior probabilities of different values of N (shown on the right of each row of Figure 6.13) provide a measure of the complexity of the model justified by the data.



(a) Image of an irregularly shaped galaxy.



(b) Image containing several galaxies.



(c) Another image containing several galaxies.

Figure 6.13: As for Figure 6.11 but fitting 32×32 images from the Hubble Space Telescope eXtreme Deep Field (Illingworth et al., 2013); each pixel has added normally distributed y-errors with $\sigma_y = 0.2$.

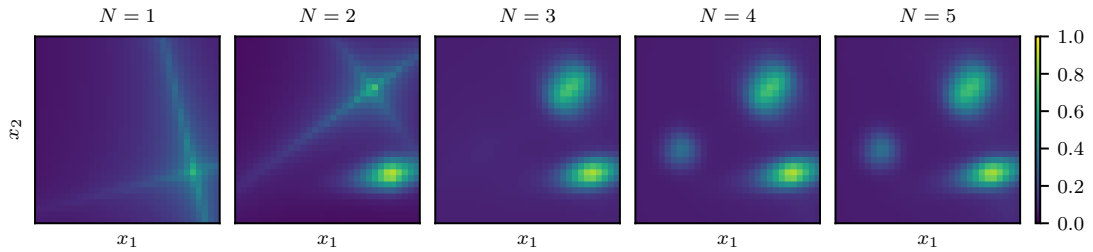


Figure 6.14: Fits of the data in Figure 6.13c conditioned on different numbers of basis functions N; these plots use results from the vanilla method.

However, unless each basis function represents a justified physical model for the sources in the image, N cannot necessarily be interpreted as the number of sources; for example a single source with a non-Gaussian structure may be represented by several Gaussian basis functions.

6.6 Conclusion

We have introduced Bayesian sparse reconstruction; a principled framework for signal reconstruction which allows the model's complexity to be determined by the data. Our approach performs well at fitting noisy 1- and 2-dimensional test data from mixture models, as well as reconstructing astronomical images.

While the techniques described in this chapter are computationally expensive (see Appendix 6.B for details of the compute used to produce our results), we show that they are now feasible in the low data regime with current software and are capable of producing excellent results. Furthermore, we intend this work to provide a proof of principle for future application of our approach to larger datasets, when advances in numerical techniques and increases in computational power make this feasible.

Appendix 6.A Code

The code used to make the results and plots in this chapter can be downloaded at `https://github.com/ejhigson/bsr` (this also includes the code used in Chapter 7).

Appendix 6.B Computational resources used

Table 6.2 shows the approximate number of core hours used for each calculation in this chapter, and is intended to provide a rough guide to the computational cost of our method. We used the CDS3 Peta4 cluster, which has 2.6GHz 16-core Intel Xeon Skylake 6142 processors (2 processors and 32 cores per node). Note that the number of core hours used can vary significantly when the same calculation is repeated.

Vanilla and adaptive calculations use `PolyChord` and dynamic adaptive calculations use `dyPolyChord`. `dyPolyChord` performs dynamic nested sampling by saving and resuming `PolyChord` runs; this is not yet parallelised in the current version of `PolyChord` and can become a bottleneck when running with large numbers of processes, increasing the amount of core hours required for this method. We intend this process to be more computationally efficient in future dynamic nested sampling software. All calculations use C++ likelihoods except the adaptive and dynamic adaptive selection of T in Figure 6.8, which were run using a Python likelihood and consequently required more computation time. The code used can be downloaded from the link in Appendix 6.A.

When fitting the same basis functions to different data sets, reconstructing more complex signals requires more computation — this can be seen in Table 6.2 for Figures 6.6a to 6.6c and Figures 6.8a to 6.8c.

Appendix 6.C Additional numerical results

This Appendix contains details of the parameters of the mixture models used to generate true signals in the numerical examples, as well as tables comparing the computational efficiency of results calculated through the adaptive and vanilla methods.

6.C.1 Parameters for test signals

Tables 6.3 to 6.5 show the parameters of the mixture models used for the signals in Figures 6.6, 6.8 and 6.11 respectively.

6.C.2 Efficiency gain results

Tables 6.6 and 6.7 show numerical values for the mean fit at the centre of the signal's domain for Figures 6.6 and 6.11, as well as estimates of the efficiency gain (6.24) from

	vanilla	adaptive	dynamic adaptive
Fitting $1d$ generalised Gaussians (Figure 6.6)			
Figure 6.6a	1	1	3
Figure 6.6b	1.5	1.5	4
Figure 6.6c	2	2	4
Fitting $1d$ tanhs (Figure 6.8)			
Figure 6.8a	1	1	3
Figure 6.8b	1.5	1.5	4
Figure 6.8c	3	3	5
Fitting $1d$ basis functions with adaptive T (Figure 6.10)			
Figure 6.10b	2	20	20
Figure 6.10a	2	20	20
Fitting $2d$ generalised Gaussians (Figures 6.11 and 6.13)			
Figure 6.11a	5	7	50
Figure 6.11b	10	14	70
Figure 6.11c	12	20	80
Figure 6.13a	8	26	70
Figure 6.13b	11	60	100
Figure 6.13c	10	40	80

Table 6.2: Approximate numbers of core hours used per calculation for results shown in this chapter; these were run on the CDS3 Peta4 cluster, which has 2.6GHz 16-core Intel Xeon Skylake 6142 processors (2 processors and 32 cores per node). For adaptive results, each calculation is a single nested sampling run. For vanilla runs a calculation involves a separate nested sampling run for each value of N — the values of the table show the total core hours used by these. For calculations fitting basis functions to 1-dimensional signals (Figures 6.6, 6.8 and 6.10) `num_repeats`=100, vanilla runs use 200 live points and adaptive runs use 1,000. For calculations fitting 2-dimensional images (Figures 6.11 and 6.13) `num_repeats`=250, vanilla runs use 400 live points and adaptive runs use 2,000. Note that plots of results all use combined inferences from 5 calculations.

# functions	a	μ	σ	β
1	0.75	0.4	0.3	2
2	0.2	0.4	0.6	5
	0.55	0.4	0.2	4
3	0.2	0.4	0.6	5
	0.35	0.6	0.07	2
	0.55	0.32	0.14	6

Table 6.3: Parameters for the sum of 1-dimensional generalised Gaussian basis functions (6.22) from which the data shown in Figure 6.6 was sampled.

# functions	a	b	w
1	0.8	0	1.5
2	0.7	-1	3
	0.9	2	-3
3	0.6	-7	8
	1	-1	3
	1.4	2	-3

Table 6.4: Parameters for the sum of 1-dimensional tanh basis functions (6.23) from which the data shown in Figure 6.8 was sampled.

# functions	a	μ_1	μ_2	σ_1	σ_2	β_1	β_2	Ω
1	0.8	0.6	0.6	0.1	0.2	2	2	$\pi/10$
2	0.5	0.5	0.4	0.4	0.2	2	2	0
	0.8	0.5	0.6	0.1	0.1	2	2	0
3	0.5	0.3	0.7	0.2	0.2	2	2	0
	0.7	0.7	0.6	0.15	0.15	2	2	0
	0.9	0.4	0.3	0.1	0.1	2	2	0

Table 6.5: Parameters for the sum of two-dimensional generalised Gaussian basis functions (6.25) from which the data shown in Figure 6.11 was sampled.

the adaptive method (with and without dynamic nested sampling) compared to the vanilla method. Efficiency gains reported use results' estimated variation, calculated from bootstrap resampling using the `nestcheck` package (Higson, 2018b).

Bootstrap resampling allows the variation of results due to the stochasticity of the nested sampling algorithm to be determined accurately without the need to perform the computation many times. However this does not include added variation due to implementation-specific effects (discussed in Chapter 4), which can lead to additional errors. These implementation-specific effects can be reduced by changing the software settings; for `PolyChord` and `dyPolyChord` this entails increasing the `num_repeats` setting and/or the number of live points. Diagnostics provided by `nestcheck` indicate the presence of such additional variation in our results; it also explains how estimates of the fit $y(0.5; \boldsymbol{\theta})$ and $y(0.5, 05; \boldsymbol{\theta})$ using different methods, in Tables 6.6 and 6.7 respectively, sometimes differ by slightly more than would be expected from their bootstrap uncertainties. As the posterior is more challenging and complex in the adaptive method than the vanilla method, this is likely to mean the efficiency gain observed in practice is lower than the estimates using the bootstrap estimates of variation with the settings we use. However we include it as a rough estimate and an indication of the efficiency gain which could be achieved with more live points and/or a higher `num_repeats` setting.

	vanilla	adaptive	dynamic adaptive
Data from 1 generalised Gaussian (shown in Figure 6.6a)			
# samples	128,719	114,507	116,867
$y(0.5;\boldsymbol{\theta})$	0.6526(4)	0.6517(3)	0.6527(1)
efficiency gain		1.7(4)	46(9)
Data from 2 generalised Gaussians (shown in Figure 6.6b)			
# samples	128,052	133,061	131,637
$y(0.5;\boldsymbol{\theta})$	0.6340(3)	0.6351(1)	0.6338(1)
efficiency gain		14(3)	4.3(9)
Data from 3 generalised Gaussians (shown in Figure 6.6c)			
# samples	152,516	171,853	173,959
$y(0.5;\boldsymbol{\theta})$	0.4040(3)	0.4029(2)	0.4072(2)
efficiency gain		2.5(5)	2.7(6)

Table 6.6: Numerical values for the accuracy of the mean fit at $x = 0.5$ using the data and nested sampling runs shown in Figure 6.6. The columns show results for the vanilla and adaptive methods using standard nested sampling and the adaptive method using dynamic nested sampling. For each data set, the first two rows show the total number of samples used by the nested sampling runs, and the mean value of $y(0.5;\boldsymbol{\theta})$. The next two rows show the efficiency gain (6.24) of the adaptive method with and without dynamic nested sampling; these are calculated using estimates of the standard deviation of results from bootstrap resampling 100 bootstrap replications. The numbers in brackets show 1σ errors on the final digit.

	vanilla	adaptive	dynamic adaptive
Data from 1 $2d$ generalised Gaussian (shown in Figure 6.11a)			
# samples	377,360	324,294	322,799
$y(0.5, 0.5; \boldsymbol{\theta})$	0.3353(3)	0.3360(2)	0.3359(1)
efficiency gain		3.7(7)	16(3)
Data from 2 $2d$ generalised Gaussians (shown in Figure 6.11b)			
# samples	476,932	520,691	503,380
$y(0.5, 0.5; \boldsymbol{\theta})$	0.6850(5)	0.6853(2)	0.6856(1)
efficiency gain		10(2)	12(2)
Data from 3 $2d$ generalised Gaussians (shown in Figure 6.11c)			
# samples	562,830	652,359	656,852
$y(0.5, 0.5; \boldsymbol{\theta})$	0.1435(1)	0.1438(1)	0.1437(1)
efficiency gain		1.2(2)	4.3(9)

Table 6.7: Numerical values for the accuracy of the mean fit at $\boldsymbol{x} = (0.5, 0.5)$ using the data and nested sampling runs shown in Figure 6.11. The columns show results for the vanilla and adaptive methods using standard nested sampling and the adaptive method using dynamic nested sampling. For each data set, the first two rows show the total number of samples used by the nested sampling runs, and the mean value of $y(0.5, 0.5; \boldsymbol{\theta})$ produced. The next two rows show the efficiency gain (6.24) of the adaptive method with and without dynamic nested sampling; these are calculated using estimates of the standard deviation of results from bootstrap resampling. The numbers in brackets show 1σ errors on the final digit.

Chapter 7

Bayesian sparse reconstruction with neural networks

We now apply our Bayesian sparse reconstruction framework introduced in Chapter 6 to artificial neural networks, where it allows a dynamic selection of the optimum network architecture. This chapter is an edited version of the latter part of Higson et al. (2019c).

7.1 Introduction

Artificial neural networks (hereafter neural networks) are a popular machine learning technique loosely inspired by biological brains. MacKay (2003, Section V) provides a good introduction; for a detailed Bayesian reference see Neal (2012). Neural networks have been successfully applied to many areas of astronomical data analysis, including to image processing (see for example Graff et al., 2014; Ball and Brunner, 2010).

Neural networks are made up of nodes ("neurons") which receive input signals and map them to a scalar signal ("activation"), which is then passed to other nodes. We restrict our analysis to "fully-connected" "feed-forward" networks, in which nodes are arranged in layers and each node receive inputs from every node the previous layer and passes its output to every node in the following layer (in this case the network is a directed acyclic graph). Layers of nodes between the network's input and output are termed "hidden layers", as their outputs are not directly specified by the signal.

Following the neural network literature, we denote the activation of the j^{th} node in the l^{th} layer as $a_j^{[l]}$; this is differentiated from the basis function amplitudes used in

Chapter 6 by the superscript label in square brackets and by the context. The activation of each node is computed as

$$a_j^{[l]} = \phi^{[l]} \left(\sum_{i=1}^{N} a_i^{[l-1]} w_{ji}^{[l]} + b_j^{[l]} \right), \tag{7.1}$$

where $w_{j1}^{[l]}, \ldots, w_{jN}^{[l]}$ are the weights assigned to the activations of the N nodes in the previous layer and conventionally an additional parameter $b_j^{[l]}$ (referred to as the "bias") is included. The activation function $\phi^{[l]}$ is typically non-linear function of the inputs such as tanh or rectifier functions.[1] For a feed-forward neural network with a d-dimension input $\boldsymbol{x}$ and one hidden layer containing N nodes, the activations are:

$$a_j^{[1]} = \phi^{[1]} \left(\sum_{i=1}^{d} x_i w_{ji}^{[1]} + b_j^{[1]} \right), \tag{7.2}$$

$$y_j = a_j^{[2]} = \phi^{[2]} \left(\sum_{i=1}^{N} a_i^{[1]} w_{ji}^{[2]} + b_j^{[2]} \right). \tag{7.3}$$

Such a network with a single output y is illustrated in Figure 7.1.

Values for the network parameters can be selected using gradient-based optimisation and regularisation — this is useful for "deep learning", in which networks have large numbers of hidden layers (are "deep") and sampling the posterior distribution over the full parameter space is not computationally feasible. Often some part of the data set is held back and used for selecting the regularisation parameter.

Bayesian methods also provide a natural framework for neural networks, and simplified Bayesian computation can be performed in the space of neural network parameter values using techniques such as Bayes by backprop (Blundell et al., 2015) and Gaussian approximations (Mackay, 1995). In addition, many regularisation techniques commonly applied to neural networks can be interpreted from a Bayesian perspective (see for example Gal, 2016).

[1] Rectifier functions such as $\phi(x) = \max(0, x)$ are now popular for deep neural networks, as they make it easier to optimise the network's weights with gradient-based methods because their gradient does not become small when x is large (Lecun et al., 2015). This chapter uses the hyperbolic tangent function as we do not rely on gradient-based optimisation and our networks only have one or two hidden layers.

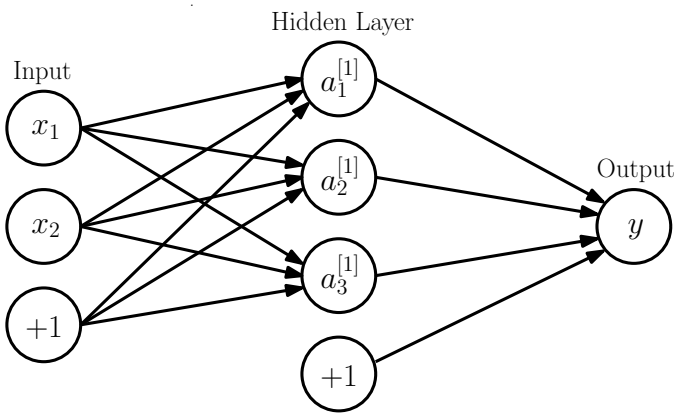


Figure 7.1: A feed-forward neural network with 2 inputs, a single hidden layer with 3 nodes and a scalar output y. Circles labelled +1 and the arrows leading from them represent the bias parameters $b_j^{[l]}$.

7.2 Applying Bayesian sparse reconstruction to neural networks

To illustrate the connection between neural networks and the basis function fitting in Chapter 6, consider fitting a scalar signal y using a signal hidden layer neural network in which the output layer has an identity activation function $\phi^{[2]}(x) = x$ and its bias parameter $b^{[2]}$ set to zero. In this case the output (7.3) is simply a sum of basis functions. If a tanh activation function is used for the nodes in the hidden layer and there is a scalar input x, such a network is equivalent to fitting 1-dimensional tanh basis functions as shown in Figures 6.8 and 6.9. In this case, determining the value of N using the framework introduced earlier in Chapter 6 represents Bayesian inference on the optimum number of nodes in the hidden layer given the data.

When there is more than one hidden layer, the output is no longer a direct sum of the inputs but our Bayesian sparse reconstruction framework can still be readily applied. Furthermore the number of hidden layers L can be determined by treating it as an integer parameter, in the same way as the basis function family was represented by the integer parameter T in Section 6.4.3. We use the same number of nodes N in each hidden layer, but if required one could allow the hidden layers to have different numbers of nodes governed by multiple integer parameters $N^{[1]}, \ldots, N^{[L]}$. We consider only a single output for simplicity, but our results easily generalises to neural networks

Parameter	Prior Type	Prior Parameters
L	Uniform (integer)	$\in \mathbb{Z} \cap [1, 2]$
N	Uniform (integer)	$\in \mathbb{Z} \cap [1, 10]$
σ_w	Uniform in σ_w^{-2} (7.5)	$\in [0.1, 10]$
output weights $w_{ij}^{[L+1]}$	Sorted Gaussian[2]	$\mu = 0, \sigma = \sigma_w$
other weights & biases	Gaussian	$\mu = 0, \sigma = \sigma_w$

Table 7.1: Priors on neural network parameters. Sorted priors have ordering enforced; see Handley et al. (2015b, Appendix A2) for more details.

with multiple outputs $(y_1, y_2, \dots)$ and to classification problems in which the output takes only discrete values.

We now apply our Bayesian sparse reconstruction framework to neural networks, and show that this approach for principled adaptive Bayesian selection of network architecture without Gaussian approximations works well for "shallow" neural networks with a small number of hidden layers. We use tanh activation functions for the nodes in the L hidden layers, and a sigmoid activation function for the output

$$\phi^{[L+1]}(x) = \text{sigmoid}(x) = \frac{1}{1 + \mathrm{e}^{-x}} = \frac{\mathrm{e}^x}{1 + \mathrm{e}^x}. \tag{7.4}$$

This conveniently maps the output y into $[0, 1]$, which is the range of the target signal in the numerical examples.

We use Gaussian priors on the neural network's weight parameters, as summarised in Table 7.1. Due to the difficulty in selecting the priors' scale *a priori*, we use a hyperparameter σ_w for the width of the Gaussian priors on the weights — this can be marginalised out when calculating posterior inferences. Following Mackay (1995) we use a uniform prior on σ_w^{-2}, meaning

$$\pi(\sigma_w) = \frac{3\sigma_w^{-3}}{\sigma_{w,\min}^{-2} - \sigma_{w,\max}^{-2}} \tag{7.5}$$

where $\sigma_w > 0$.

[2]For neural networks with only one hidden layer, following Mackay (1995), priors on the weights leading to the output are further restricted to only be non-zero in the positive half of the Gaussian. This exploits a symmetry in the parameter space as $\tanh(x)$ is symmetric under changes of sign in x.

7.3 Fitting 2-dimensional images with neural networks

Figure 7.2 shows signal reconstruction with neural networks, including Bayesian inference on the number of hidden layers L and nodes per hidden layer N, using our Bayesian sparse reconstruction framework. Readers who are less familiar with neural networks might expect them to struggle to fit the challenging data set (the same one used in Figure 6.11c) using their tanh activation functions. However we see that our approach yields good results, and the network is able to reconstruct the generalised Gaussians in the signal by overlaying 2-dimensional tanh functions from different nodes. How it does this is illustrated in Figure 7.3, which shows fits conditioned on different values of L and N. The first two rows with $L = 1$ represent a network with a single hidden layer, and show how increasing N allows the tanh functions to first create a triangle around the three maxima and then to represent the maxima themselves. The second two rows use $L = 2$; a comparison with the $L = 1$ plots shows how the two hidden layer architecture allows more complex signal structure to be represented using a given value of N. The posterior distribution of L heavily favours two hidden layers, with the adaptive method using dynamic nested sampling giving $P(L = 2|\mathcal{L}, \pi) = 0.984 \pm 0.007$.

The network with $L = 2$ hidden layers and $N = 10$ nodes per hidden layer has 151 weight parameters plus the hyperparameter σ_w and the integer parameters L and N; the resulting parameter space is 154-dimensional, as well as highly multimodal and degenerate. The default `PolyChord` and `dyPolyChord` settings for this dimensionality are $25 \times d = 3,850$ live points and $5 \times d = 770$, so it is not surprising that with the settings used our results show large inconsistencies in the calculated posterior distribution of L and N due to implementation-specific effects (see Chapter 4 for a detailed discussion). However our approach is still able to allocate almost all the posterior mass to L, N combinations which are good fits for the data, leading to good results and demonstrating the robustness of the method.

Furthermore, neural network and basis function fits can be compared using the adaptive method. For example one could include an additional integer parameter T, with values $T = 1$ and $T = 2$ representing fitting with 2-dimensional generalised Gaussians and with neural networks respectively.

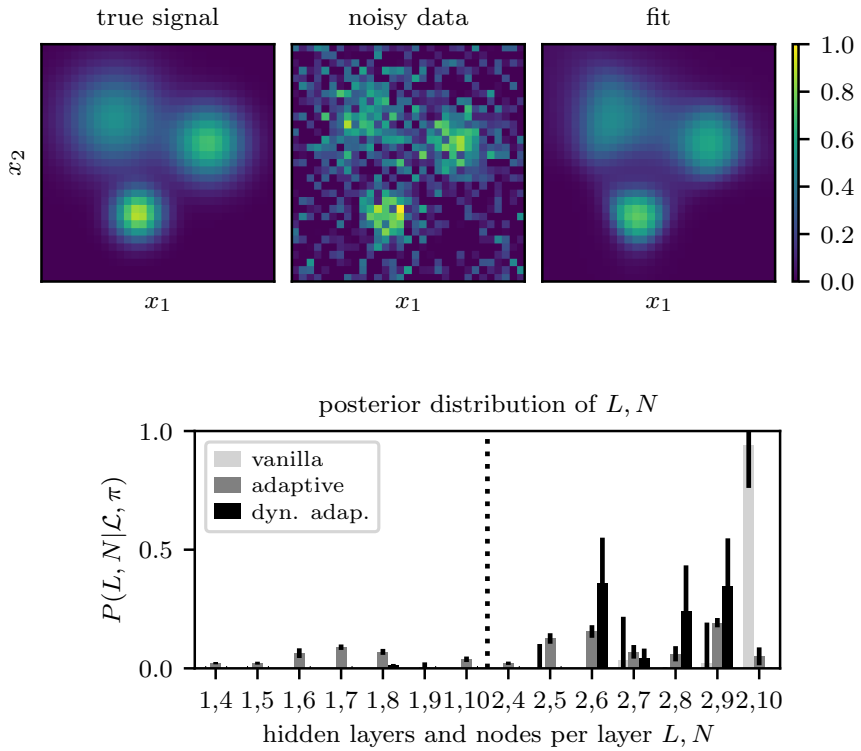


Figure 7.2: Fitting neural networks with the number of hidden layers L and nodes per hidden layer N determined through Bayesian inference. The first two colour plots show the true signal and the data, which includes added normally distributed y-errors with $\sigma_y = 0.2$; these are the same as in Figure 6.11c. The third colour plot shows the mean value of $y(\boldsymbol{x})$ from the posterior samples produced using the adaptive method with dynamic nested sampling. The bar plot displays the posterior distribution on L, N; values calculated using the vanilla method and the adaptive method without nested sampling are also included for comparison. Bars showing posterior probabilities for $N = 1$, $N = 2$ and $N = 3$ are omitted for brevity as they contain negligible posterior mass for both $L = 1$ and $L = 2$. Adaptive results use a combined inference from 5 runs, each of which computes a full posterior on L, N and uses 2,000 live points; adaptive runs using dynamic nested sampling have `dyPolyChord` settings $n_{\text{init}} = 1,000$ and `dynamic_goal` $= 1$. Results for the vanilla method use 5 separate runs, each with 400 live points, to compute the evidence for each combination L, N. All runs use the setting `num_repeats` $= 250$

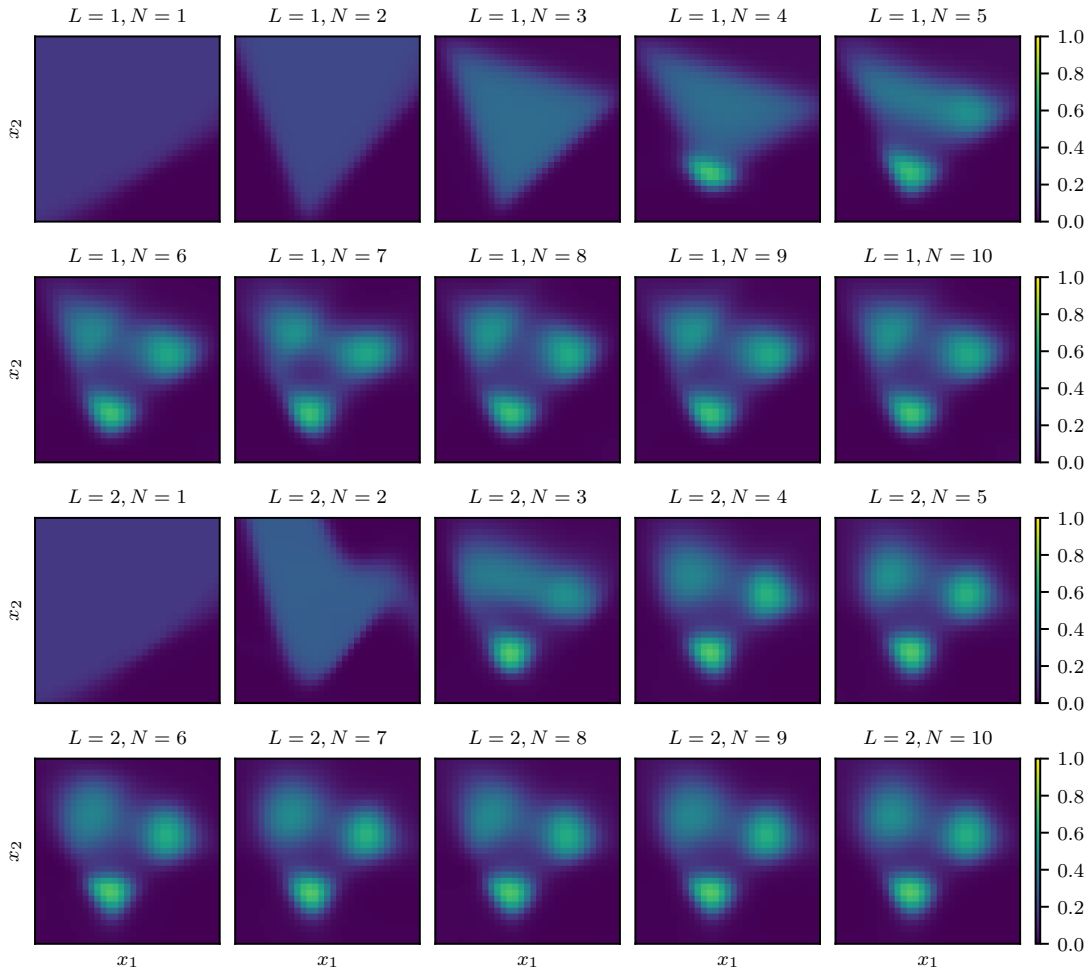


Figure 7.3: Fits from Figure 7.2 conditioned on different numbers of hidden layers L and nodes per hidden layer N. The plots use results from the vanilla method.

7.4 Application to astronomical images

We now apply neural networks to the Hubble Space Telescope eXtreme Deep Field images used in Section 6.5.1. We find the adaptive selection of L for these data sets strongly favours $L = 2$ over $L = 1$, so for simplicity we show only results using 2 hidden layers.

Figure 7.4 shows results from fitting neural networks with 2 hidden layers to the data used in Figure 6.13, with fits conditioned on specific values of N shown in Figure 7.5. As for the 2-dimensional Gaussian basis functions, a visual assessment shows the neural networks are able to faithfully reconstruct the true image from the noisy data with

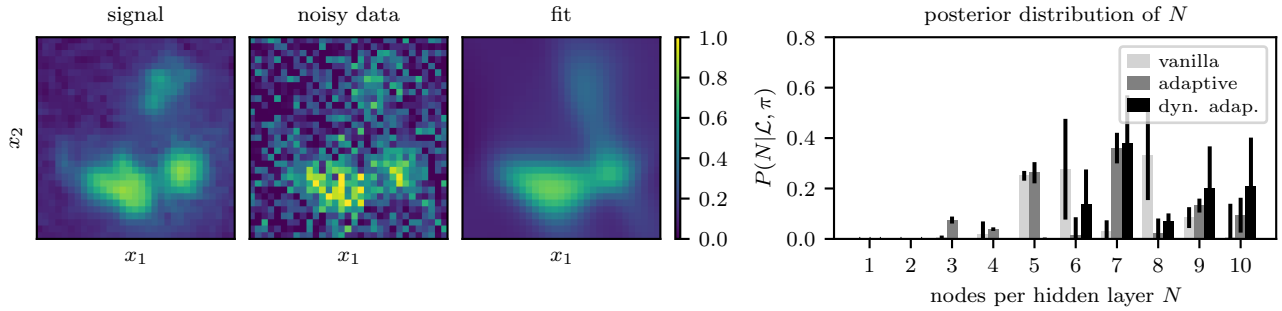
good accuracy. However the neural networks (with tanh activation functions) do not provide as natural a representation of the blob-shaped sources as the 2-dimensional generalised Gaussians, so the fits are not as good as those shown in Figures 6.13 and 6.14. Nevertheless the example provides a proof of principle, and the versatility of neural networks means they can be applied to a wide range of data sets using this technique.

The posterior distributions of the number of nodes in each hidden layer N, shown on the right of each row of plots in Figure 7.4, illustrate the number of nodes (degree of complexity of the model) which is justified by the astronomical image. However, unlike the basis functions, the contributions of each individual node to the output fit is not readily interpretable. As in the previous section, we find that the network's fits of the images are good — despite inconsistencies in the posterior probabilities of different values of N between the different methods due to implementation-specific effects. The posterior distribution of N can be calculated more precisely using more computational resources (for example by increasing `PolyChord` and `dyPolyChord`'s `num_repeats` settings).

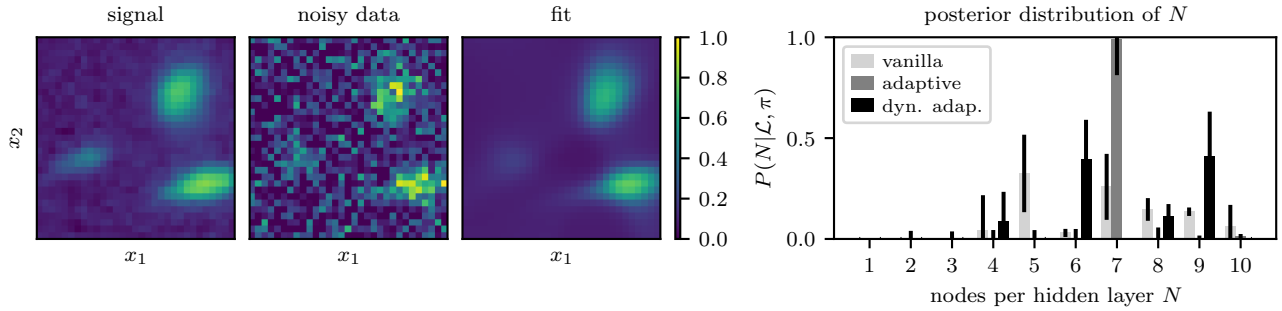
7.5 Conclusion

We have demonstrated the application of the Bayesian sparse reconstruction framework to neural networks — a popular and versatile machine learning technique. Our approach allows Bayesian inference to be performed over the space of network architectures in a principled Bayesian manner; this has many possible uses in astronomy and beyond. One appealing possible application is to autoencoders; a type of neural network which is used to learn an efficient representation of a signal (for more details and an astronomical application, see Graff et al., 2014). In this context, Bayesian sparse reconstruction will allow the architecture of the network used to encode a signal to be determined by the data.

As with the basis function fits in Chapter 6, the examples in this chapter are intended as a proof of principle. Future advances in computational hardware and numerical techniques will allow this approach to neural network fitting to be applied to larger and more complex data sets.



(a) Image containing several galaxies.



(b) Another image containing several galaxies.

Figure 7.4: Fitting 32×32 images from the Hubble Space Telescope eXtreme Deep Field (Illingworth et al., 2013) using neural networks with two hidden layers. In each row the 2 plots on the left show the true signal and the data, which includes added normally distributed y-errors with $\sigma_y = 0.2$. The third column shows the mean value of y from the posterior samples produced using the adaptive method with dynamic nested sampling. The bar plots display the posterior distribution for different numbers of nodes per hidden layer N; values calculated using the vanilla method and the adaptive method without dynamic nested sampling are also included for comparison. Adaptive results show a combined inference form 5 runs, each of which computes a full posterior on N and uses 2,000 live points; adaptive runs using dynamic nested sampling have `dyPolyChord` settings $n_{\text{init}} = 1,000$ and `dynamic_goal` $= 1$. Results for the vanilla method use separate runs, each with 400 live points, to compute the evidence for each value of N. All runs use the setting `num_repeats` $= 250$.

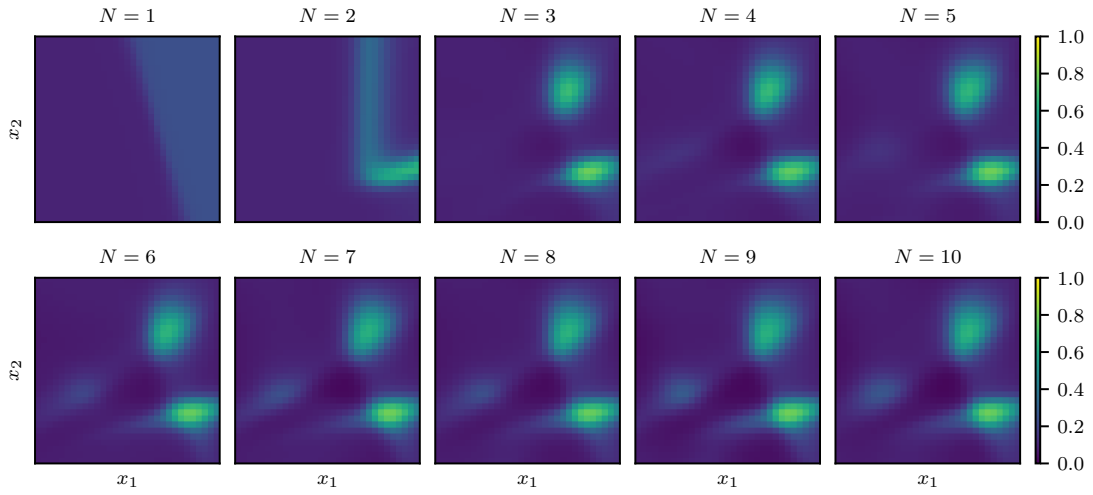


Figure 7.5: Fits of the data sets shown in Figure 7.4b conditioned on different numbers of nodes per hidden layer N. The plots use results from the vanilla method.

Appendix 7.A Code

The code used to make the results and plots in this chapter can be downloaded at `https://github.com/ejhigson/bsr` (this also includes the code used in Chapter 6).

Appendix 7.B Computational resources used

Table 7.2 shows the approximate number of core hours used for each calculation in this chapter, and is intended to provide a rough guide to the computational cost of our method. We used the CDS3 Peta4 cluster, which has 2.6GHz 16-core Intel Xeon Skylake 6142 processors (2 processors and 32 cores per node). Note that the number of core hours used can vary significantly when the same calculation is repeated.

Vanilla and adaptive calculations use `PolyChord` and dynamic adaptive calculations use `dyPolyChord`; all results in this chapter use C++ likelihoods. `dyPolyChord` performs dynamic nested sampling by saving and resuming `PolyChord` runs; this is not yet parallelised in the current version of `PolyChord` and can become a bottleneck when running with large numbers of processes, increasing the amount of core hours required for this method. We intend this process to be more computationally efficient in future dynamic nested sampling software.

	vanilla	adaptive	dynamic adaptive
Fitting neural networks with adaptive L (Figure 7.2)			
Figure 7.2	150	200	300
Fitting neural networks with 2 hidden layers (Figure 7.4)			
Figure 7.4a	100	100	200
Figure 7.4b	100	100	200

Table 7.2: Approximate numbers of core hours used per calculation for the neural network fits shown Figures 7.2 and 7.4; these were run on the CDS3 Peta4 cluster, which has 2.6GHz 16-core Intel Xeon Skylake 6142 processors (2 processors and 32 cores per node). For adaptive results, each calculation is a single nested sampling run. For vanilla runs a calculation involves a separate nested sampling run for each value of N — the values of the table show the total core hours used by these. Vanilla runs use 400 live points and adaptive runs use 2,000; all calculations used `num_repeats`=250. Note that plots of results all use combined inferences from 5 calculations.

Chapter 8

Conclusion

We now conclude this thesis by reviewing the work presented and suggesting possible areas of future research.

Chapter 3 analysed sampling errors in nested sampling parameter estimation, and introduced a method for estimating the sampling errors from the output of a single nested sampling run. This was then used in Chapter 4 to create diagnostic tests for assessing whether or not software had performed the nested sampling algorithm accurately. Given the popularity of nested sampling in astronomy, these methods can be used in future research in a wide range of areas (example applications involving gravitational waves and *Planck* survey data are given in Sections 3.7 and 4.7). In addition to the material contained in this thesis, a significant part of the contribution of this work is the open-source software package `nestcheck` (Higson, 2018b). This provides well-tested implementations of these methods for use by researchers, and is actively maintained and improved. In particular, the diagnostic tests in Chapter 4 are — to the best of our knowledge — the first such tests for implementation-specific effects in nested sampling runs which can be applied to practical problems. Given the challenges of detecting these effects, more work is needed to further test our methods in current research applications and refine them — or produce better alternatives.

The work in Chapters 3 and 4 grew from our efforts to understand the sampling errors present when using the adaptive method in earlier versions of the Bayesian sparse reconstruction framework (presented in Chapter 6), and our primary goal was to be able to numerically estimate uncertainties for calculations used in our astrophysics research. While this might seem sufficient from the perspective of a physicist, further theoreti-

cal work is also needed to improve the understanding of nested sampling's statistical properties to the level of most alternative numerical methods. Salomone et al. (2018) cites the lack of understanding of the distinctive errors in parameter estimation and due to implementation-specific effects as two important reasons why nested sampling is not more popular with statisticians; Chapters 3 and 4 make some contribution to this but are more focused on practical applications than theoretical proofs. In our opinion the degree to which this highly general technique for Bayesian computation is principally used by physicists (and in particular astrophysicists) is somewhat anomalous, and is likely due in large part to its historic development by astrophysicists. More theoretical work and interdisciplinary collaboration on nested sampling will help contribute to a more widespread adoption of the technique, and has the potential to benefit all concerned.

Dynamic nested sampling, introduced in Chapter 5, grew directly from our efforts to understand the sources of sampling error in nested sampling parameter estimation (presented in Chapter 3). The algorithm offers order-of-magnitude increases in computational efficiency over standard nested sampling for many problems, in particular for high-dimensional parameter estimation. Dynamic nested sampling has been applied to a variety of problems in astrophysics (see for example Orazio et al., 2018; Guillochon et al., 2018; Zucker et al., 2018), and has the potential to be widely used for applications in astronomy and beyond. For this to happen, the most important target of future research is the development of next-generation dynamic nested sampling software. `dyPolyChord` (Higson, 2018a) provides the current state-of-the-art nested sampling computational performance for many problems, but is limited by certain aspects of `PolyChord` (which was designed before the creation of the dynamic nested sampling algorithm). These include relatively high computational costs of saving, loading and resuming runs. `PolyChord 2`, which is currently in development, will be able to handle problems with $\sim 1,000$ dimensions and is designed to incorporate dynamic nested sampling — allowing large increases in efficiency. `PolyChord 2` will also be optimised for fast saving and resuming of runs; we estimate that for many practical problems resuming the nested sampling process will have a lower computational cost than a single likelihood call. In addition to its implementation, future research could allow improvements to be made to the dynamic nested sampling algorithm; tuning for a specific parameter estimation problem (discussed in Appendix 5.D) is one promising possibility.

The second theme of this thesis is sparse reconstruction of noisy signals. Chapter 6 introduced our Bayesian sparse reconstruction methodology, in which the signal is reconstructed by basis functions whose number and (if needed) form are determined by the data themselves. This principled Bayesian approach also provides a natural framework for reinterpreting conventional sparse reconstruction and regularisation techniques. We showed our approach is feasible with current numerical methods and that, in addition to being philosophically appealing, it is also capable of producing excellent reconstructions from challenging data. Furthermore, by using the adaptive method, the number and type of basis functions to be used in the signal reconstruction can be treated as an integer parameters and the calculation of posterior odds can be performed by parameter estimation. This offers large potential increases in computational efficiency and can be effectively combined with dynamic nested sampling. We successfully applied our approach to 2-dimensional image reconstruction, including of astronomical images from the Hubble Space Telescope eXtreme Deep Field (Illingworth et al., 2013).

Future research should aim to apply the Bayesian sparse reconstruction framework to more complex and varied astronomical applications, and should include quantitative tests of the quality of the fit. Additional applications include fitting vector signals $\boldsymbol{y}(\boldsymbol{x})$, such as colour images — where each pixel can be expressed as a 3-dimensional signal with red, green and blue intensities. Another possibility is to apply the method to deconvolution. Since more complex applications will be more computationally challenging, future research in this area will also be greatly helped by advances in numerical methods such as next-generation dynamic nested sampling software.

In Chapter 7 we showed our Bayesian sparse reconstruction framework also naturally applies to neural networks, and allows Bayesian inference over the space of possible network architectures by treating the number of nodes and hidden layers as parameters. The technique was demonstrated using challenging 2-dimensional image reconstruction problems. Like the basis function reconstructions in Chapter 6, this principled approach to neural networks has a wide variety of potential uses; a particularly appealing possibility is applying the method to autoencoders (mentioned in Section 7.5).

This thesis has presented work on two connected themes: advances in nested sampling and Bayesian sparse reconstruction of signals. These are timely topics of research given the strong current interest in Bayesian astrostatistics, and the continued growth in the quantity of data and computational power available to astronomers. We hope

this work will make some modest contribution to meeting the challenges posed by future astronomical data sets.

Bibliography

J. G. Ables. Maximum Entropy Spectral Analysis. *Astronomy and Astrophysics Supplement*, 15:383–393, 1974.

S. Ahn and J. Fessler. Standard Errors of Mean, Variance, and Standard Deviation Estimators. *EECS Department, University of Michigan*, pages 1–2, 2003.

S. Aitken and O. E. Akman. Nested sampling for parameter inference in systems biology: application to an exemplar circadian model. *BMC systems biology*, 7:72, 2013. doi: 10.1186/1752-0509-7-72.

H. Akaike. A New Look at the Statistical Model Identification. *IEEE Transactions on Automatic Control*, 19(6):716–723, 1974. doi: 10.1109/TAC.1974.1100705.

R. Allison and J. Dunkley. Comparison of sampling techniques for Bayesian parameter estimation. *Monthly Notices of the Royal Astronomical Society*, 437(4):3918–3928, 2014. doi: 10.1093/mnras/stt2190.

C. Andrieu and J. Thoms. A tutorial on adaptive MCMC. *Statistics and Computing*, 18(4):343–373, 2008. doi: 10.1007/s11222-008-9110-y.

F. Bach, R. Jenatton, J. Mairal, and G. Obozinski. Optimization with Sparsity-Inducing Penalties. *Foundations and Trends in Machine Learning*, 4(1):1–106, 2012. doi: 10.1561/2200000015.

F. Bacon. Novum Organum Scientiarum, 1620.

N. M. Ball and R. J. Brunner. Data Mining and Machine Learning in Astronomy. *International Journal of Modern Physics D*, 19(07):1049–1106, 2010. doi: 10.1142/S0218271810017160.

L. Barack and C. Cutler. Using LISA extreme-mass-ratio inspiral sources to test off-Kerr deviations in the geometry of massive black holes. *Physical Review D*, 75(4): 1–12, 2007. doi: 10.1103/PhysRevD.75.042003.

T. Bayes and R. Price. An Essay Towards Solving a Problem in the Doctrines of Chances. *Philosophical Transactions of the Royal Society of London*, 53:370–418, 1763. doi: 10.1093/biomet/45.3-4.293.

F. Beaujean and A. Caldwell. Initializing adaptive importance sampling with Markov chains. *arXiv preprint arXiv:1304.7808*, 2013.

M. Betancourt. Nested sampling with constrained Hamiltonian Monte Carlo. In *AIP Conference Proceedings*, volume 1305, pages 165–172, 2011. ISBN 9780735408609. doi: 10.1063/1.3573613.

M. Betancourt. A Conceptual Introduction to Hamiltonian Monte Carlo. *arXiv preprint arXiv:1701.02434*, 2017.

I. Bezáková, D. Stefankovic, V. V. Vazirani, and E. Vigoda. Accelerating simulated annealing for the permanent and combinatorial counting problems. *SIAM*, 37(5): 1429–1454, 2008. doi: 10.1137/050644033.

D. M. Blei, A. Kucukelbir, and J. D. McAuliffe. Variational Inference: A Review for Statisticians. *Journal of the American Statistical Association*, 112(518):859–877, 2017. doi: 10.1080/01621459.2017.1285773.

C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra. Weight Uncertainty in Neural Networks. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 1613–1622. PMLR, 2015.

J. Bobin, J. L. Starck, and R. Ottensamer. Compressed sensing in astronomy. *IEEE Journal on Selected Topics in Signal Processing*, 2(5):718–726, 2008. doi: 10.1109/JSTSP.2008.2005337.

B. J. Brewer and D. Foreman-Mackey. DNest4: Diffusive Nested Sampling in C++ and Python. *Journal of Statistical Software, Articles*, 86(7):1–33, 2018. doi: 10.18637/jss.v086.i07.

B. J. Brewer, L. B. Pártay, and G. Csányi. Diffusive nested sampling. *Statistics and Computing*, 21(4):649–656, 2011. doi: 10.1007/s11222-010-9198-8.

J. Buchner. A statistical test for Nested Sampling algorithms. *Statistics and Computing*, 26(1-2):383–392, 2016. doi: 10.1007/s11222-014-9512-y.

X. Cai, M. Pereyra, and J. D. McEwen. Uncertainty quantification for radio interferometric imaging - I. Proximal MCMC methods. *Monthly Notices of the Royal Astronomical Society*, 480(3):4154–4169, 2018. doi: 10.1093/mnras/sty2004.

E. Cameron and A. Pettitt. Recursive Pathways to Marginal Likelihood Estimation with Prior-Sensitivity Analysis. *Statistical Science*, 29(3):397–419, 2014. doi: 10.1214/13-STS465.

E. Candès, J. Romberg, and T. Tao. Robust uncertainty principles: Exact signal reconstruction from highly Incomplete Frequency Information. *IEEE Transactions on Information Theory*, 52(2):489–509, 2006a. doi: 10.1109/TIT.2005.862083.

E. Candès, J. Romberg, and T. Tao. Stable signal recovery from incomplete and inaccurate measurements. *Communications on Pure and Applied Mathematics*, 59(8):1207–1223, 2006b. doi: 10.1002/cpa.20124.

N. Chopin and C. P. Robert. Properties of nested sampling. *Biometrika*, 97(3):741–755, 2010. doi: 10.1093/biomet/asq021.

A. J. K. Chua, S. Hee, W. J. Handley, E. Higson, C. J. Moore, J. R. Gair, M. P. Hobson, and A. N. Lasenby. Towards a framework for testing general relativity with extreme-mass-ratio-inspiral observations. *Monthly Notices of the Royal Astronomical Society*, 478(1):28–40, 2018. doi: 10.1093/mnras/sty1079.

M. K. Cowles and B. P. Carlin. Markov chain Monte Carlo convergence diagnostics: a comparative review. *Journal of the American Statistical Association*, 91(434):883–904, 1996.

P. Del Moral, A. Doucet, and A. Jasra. Sequential Monte Carlo samplers. *Journal of the Royal Statistical Society. Series B: Statistical Methodology*, 68(3):411–436, 2006. doi: 10.1111/j.1467-9868.2006.00553.x.

D. L. Donoho. Compressed sensing. *IEEE Transactions on Information Theory*, 52(4): 1289–1306, 2006. doi: 10.1109/TIT.2006.871582.

C. Doss, J. Flegal, G. Jones, and R. Neath. Markov chain monte carlo estimation of quantiles. *Electronic Journal of Statistics*, 8:2448–2478, 2015. doi: 10.1214/14-EJS957.

D. J. Earl and M. W. Deem. Parallel tempering: Theory, applications, and new perspectives. *Physical Chemistry Chemical Physics*, 7(23):3910–3916, 2005. doi: 10.1039/b509983h.

B. Efron. Bootstrap Methods: Another Look at the Jackknife. *The Annals of Statistics*, 7(1):1–26, 1979. doi: 10.1214/aos/1176344552.

B. Efron. Why isn't everyone a Bayesian? *American Statistician*, 40(1):1–5, 1986. doi: 10.1080/00031305.1986.10475342.

B. Efron and R. Tibshirani. Bootstrap Methods for Standard Errors, Confidence Intervals, and Other Measures of Statistical Accuracy. *Statistical Science*, 1(1):54–77, 1986. doi: 10.1214/ss/1177013815.

M. Elad, B. Matalon, J. Shtok, and M. Zibulevsky. A Wide-Angle View at Iterated Shrinkage Algorithms. *Proc. SPIE, Wavelets XII*, 6701:1–19, 2007. doi: 10.1117/12.741299.

Y. Eldar and G. Kutyniok. *Compressed Sensing: Theory and Applications*. Cambridge University Press, 2012. ISBN 1107005582. doi: 10.1017/cbo9780511794308.

M. Evans. Discussion of Nested sampling for Bayesian computations by John Skilling. *Bayesian Statistics*, 8:491–524, 2007.

G. Fasano and A. Franceschini. A multidimensional version of the Kolmogorov-Smirnov test. *Monthly Notices of the Royal Astronomical Society*, 225:155–170, 1987. doi: 10.1007/s10342-011-0499-z.

F. Feroz and M. P. Hobson. Multimodal nested sampling: An efficient and robust alternative to Markov Chain Monte Carlo methods for astronomical data analyses. *Monthly Notices of the Royal Astronomical Society*, 384(2):449–463, 2008. doi: 10.1111/j.1365-2966.2007.12353.x.

F. Feroz, M. P. Hobson, and M. Bridges. MultiNest: an efficient and robust Bayesian inference tool for cosmology and particle physics. *Monthly Notices of the Royal Astronomical Society*, 398(4):1601–1614, sep 2008. doi: 10.1111/j.1365-2966.2009.14548.x.

F. Feroz, M. P. Hobson, E. Cameron, and A. N. Pettitt. Importance Nested Sampling and the MultiNest Algorithm. *arXiv preprint arXiv:1306.2144*, 2013.

F. Feroz. *Bayesian Methods for Astrophysics and Particle Physics*. PhD thesis, University of Cambridge, 2008.

F. Feroz and J. Skilling. Exploring multi-modal distributions with nested sampling. In *AIP Conference Proceedings*, volume 1553, pages 106–113, 2013. ISBN 9780735411791. doi: 10.1063/1.4819989.

J. M. Flegal, M. Haran, and G. L. Jones. Markov Chain Monte Carlo: Can We Trust the Third Significant Figure? *Statistical Science*, 23(2):250–260, 2008. doi: 10.1214/08-STS257.

D. Foreman-Mackey, D. W. Hogg, D. Lang, and J. Goodman. emcee: The MCMC Hammer. *Publications of the Astronomical Society of the Pacific*, 125:306–312, 2013. doi: 10.1086/670067.

N. Friel and J. Wyse. Estimating the evidence - a review. *Statistica Neerlandica*, 66(3): 288–308, 2012. doi: 10.1111/j.1467-9574.2011.00515.x.

N. Friel, M. Hurn, and J. Wyse. Improving power posterior estimation of statistical evidence. *Statistics and Computing*, 24(5):709–723, 2014. doi: 10.1007/s11222-013-9397-1.

J. R. Gair and N. Yunes. Approximate Waveforms for Extreme-Mass-Ratio Inspirals in Modified Gravity Spacetimes. *Physical Review D*, 84(6), 2011. doi: 10.1103/PhysRevD.84.064016.

J. R. Gair, M. Vallisneri, S. L. Larson, and J. G. Baker. Testing general relativity with low-frequency, space-based gravitational-wave detectors. *Living Reviews in Relativity*, 16, 2013. doi: 10.12942/lrr-2013-7.

Y. Gal. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learning*, volume 48, 2016.

A. Gelman. Objections to Bayesian statistics. *Bayesian Analysis*, 3(3):445–450, 2008. doi: 10.1214/08-BA318.

A. Gelman and X.-L. Meng. Simulating normalizing constants: from importance sampling to bridge sampling to path sampling. *Statistical Science*, 13(2):163–185, 1998. doi: 10.1214/ss/1028905934.

S. Geman and D. Geman. Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6(6):721–741, 1984. doi: 10.1109/TPAMI.1984.4767596.

J. Goodman and J. Weare. Ensemble samplers with affine invariance. *Communications in Applied Mathematics and Computational Science*, 5(1):65–80, 2010. doi: 10.2140/camcos.2010.5.65.

P. Graff, F. Feroz, M. P. Hobson, and A. Lasenby. SKYNET: An efficient and robust neural network training tool for machine learning in astronomy. *Monthly Notices of the Royal Astronomical Society*, 441(2):1741–1759, 2014. doi: 10.1093/mnras/stu642.

P. J. Green. Reversible Jump Markov Chain Monte Carlo Computation and Bayesian Model Determination. *Biometrika*, 82(4):711–732, 1995. doi: 10.1093/biomet/82.4.711.

J. Guillochon, M. Nicholl, V. A. Villar, B. Mockler, G. Narayan, K. S. Mandel, E. Berger, and P. K. G. Williams. MOSFiT: Modular Open-Source Fitter for Transients. *The Astrophysical Journal Supplement Series*, 236(1):6, 2018. doi: 10.3847/1538-4365/aab761.

S. F. Gull and G. J. Daniell. Image reconstruction from incomplete and noisy data. *Nature*, 272(5655):686–690, 1978. doi: 10.1038/272686a0.

J. Hadamard. Sur les Problemes aux Derivees Partielles et Leur Signification Physique. *Princeton University Bulletin*, 13:49–52, 1902.

W. Handley, M. Hobson, and A. Lasenby. PolyChord: Nested sampling for cosmology. *Monthly Notices of the Royal Astronomical Society: Letters*, 450(1):L61–L65, 2015a. doi: 10.1093/mnrasl/slv047.

W. Handley, M. Hobson, and A. Lasenby. PolyChord: next-generation nested sampling. *Monthly Notices of the Royal Astronomical Society*, 15:1–15, 2015b. doi: 10.1093/mnras/stv1911.

W. Handley. fgivenx: A Python package for functional posterior plotting. *Journal of Open Source Software*, 3(28):849, 2018. doi: 10.21105/joss.00849.

W. J. Handley and M. Millea. Maximum entropy priors with derived parameters in a specified distribution. *arXiv preprint arXiv:1804.08143*, 2018.

W. K. Hastings. Monte Carlo Sampling Methods Using Markov Chains and Their Applications. *Biometrika*, 57(1):97–109, 1970. doi: 10.2307/2334940.

S. Hee, W. Handley, M. Hobson, and A. Lasenby. Bayesian model selection without evidences: Application to the dark energy equation-of-state. *Monthly Notices of the Royal Astronomical Society*, 455(3):2461–2473, 2016. doi: 10.1093/mnras/stv2217.

S. Hee, J. Vázquez, W. Handley, M. Hobson, and A. Lasenby. Constraining the dark energy equation of state using Bayes' theorem and the Kullback-Leibler divergence. *Monthly Notices of the Royal Astronomical Society*, 466(1):369–377, 2017. doi: 10.1093/mnras/stw3102.

E. Higson. dyPolyChord: dynamic nested sampling with PolyChord. *Journal of Open Source Software*, 3(29):965, 2018a. doi: 10.21105/joss.00965.

E. Higson. nestcheck: error analysis, diagnostic tests and plots for nested sampling calculations. *Journal of Open Source Software*, 3(29):916, 2018b. doi: 10.21105/joss.00916.

E. Higson. perfectns: perfect dynamic and standard nested sampling for spherically symmetric likelihoods and priors. *Journal of Open Source Software*, 3(30):985, 2018c. doi: 10.21105/joss.00985.

E. Higson, W. Handley, M. Hobson, and A. Lasenby. Sampling errors in nested sampling parameter estimation. *Bayesian Analysis*, 13(3):873–896, 2018. doi: 10.1214/17-BA1075.

E. Higson, W. Handley, M. Hobson, and A. Lasenby. Dynamic nested sampling: an improved algorithm for parameter estimation and evidence calculation. *Statistics and Computing*, 2019a. doi: 10.1007/s11222-018-9844-0.

E. Higson, W. Handley, M. Hobson, and A. Lasenby. nestcheck: diagnostic tests for nested sampling calculations. *Monthly Notices of the Royal Astronomical Society*, 483 (2):2044–2056, 2019b. doi: 10.1093/mnras/sty3090.

E. Higson, W. Handley, M. Hobson, and A. Lasenby. Bayesian sparse reconstruction: a brute-force approach to astronomical imaging and machine learning. *Monthly Notices of the Royal Astronomical Society*, 483(4):4828–4846, 2019c. doi: 10.1093/mnras/sty3307.

M. Hobson, A. Jones, A. Lasenby, and F. Bouchet. Foreground separation methods for satellite observations of the cosmic microwave background. *Monthly Notices of the Royal Astronomical Society*, 300:1–29, 1998. doi: 10.1046/j.1365-8711.1998.01777.x.

A. E. Hoerl and R. W. Kennard. Ridge Regression: Biased Estimation for Nonorthogonal Problems. *Technometrics*, 12(1):55–67, 1970. doi: 10.1080/00401706.1970.10488634.

M. D. Hoffman and A. Gelman. The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo. *Journal of Machine Learning Research*, 15(1): 1593–1623, 2014.

D. W. Hogg and D. Foreman-Mackey. Data Analysis Recipes: Using Markov Chain Monte Carlo. *The Astrophysical Journal Supplement Series*, 236(1):11, 2018. doi: 10.3847/1538-4365/aab76e.

M. Huber and S. Schott. Random construction of interpolating sets for high-dimensional integration. *Journal of Applied Probability*, 51(1):92–105, 2014. doi: 10.1239/jap/1395771416.

S. Hug, M. Schwarzfischer, J. Hasenauer, C. Marr, and F. J. Theis. An adaptive scheduling scheme for calculating Bayes factors with thermodynamic integration using Simpson's rule. *Statistics and Computing*, 26(3):663–677, 2016. doi: 10.1007/s11222-015-9550-0.

G. D. Illingworth, D. Magee, P. a. Oesch, R. J. Bouwens, I. Labbé, M. Stiavelli, P. G. van Dokkum, M. Franx, M. Trenti, C. M. Carollo, and V. Gonzalez. the Hst Extreme Deep Field (Xdf): Combining All Acs and Wfc3/Ir Data on the Hudf Region Into the Deepest Field Ever. *The Astrophysical Journal Supplement Series*, 209(1):6, 2013. doi: 10.1088/0067-0049/209/1/6.

Z. Ivezić, A. Connolly, J. VanderPlas, and A. Gray. *Statistics, Data Mining, and Machine Learning in Astronomy*. Princeton University Press, 2014. ISBN 9788578110796. doi: 10.1088/1751-8113/44/8/085201.

H. Jeffreys. *Theory of probability*. Oxford University Press, London, 1961.

S. Ji, Y. Xue, and L. Carin. Bayesian compressive sensing. *IEEE Transactions on Signal Processing*, 56(6):2346–2356, 2008. doi: 10.1109/TSP.2007.914345.

R. W. Johnson. An Introduction to the Bootstrap. *Teaching Statistics*, 23(2):49–54, 2001. doi: 10.1111/1467-9639.00050.

D. M. Jones and A. F. Heavens. Bayesian photometric redshifts of blended sources. *Monthly Notices of the Royal Astronomical Society*, 483(2):2487–2505, 2018. doi: 10.1093/mnras/sty3279.

C. R. Keeton. On statistical uncertainty in nested sampling. *Monthly Notices of the Royal Astronomical Society*, 414(2):1418–1426, 2011. doi: 10.1111/j.1365-2966.2011.18474.x.

S. Kirkpatrick, C. Gelatt, and M. Vecchi. Optimization by Simulated Annealing. *Science*, 220(4598):671–680, 1983. doi: 10.1126/science.220.4598.671.

A. N. Lasenby, R. B. Barreiro, and M. P. Hobson. Regularization and Inverse Problems. In *Mining the Sky*, pages 15–32. Springer, 2001. doi: 10.1007/10849171_2.

Y. Lecun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015. doi: 10.1038/nature14539.

A. M. Legendre. *Nouvelles methodes pour la determination des orbites des cometes*. F. Didot, 1805.

A. Lewis. Efficient sampling of fast and slow cosmological parameters. *Physical Review D*, 87(10):103529, 2013. doi: 10.1103/PhysRevD.87.103529.

A. Lewis and S. Bridle. Cosmological parameters from CMB and other data: a Monte-Carlo approach. *Physical Review D*, 66(10):103511, 2002. doi: 10.1103/PhysRevD.66.103511.

A. Lewis, A. Challinor, and A. Lasenby. Efficient Computation of Cosmic Microwave Background Anisotropies in Closed Friedmann-Robertson-Walker Models. *The Astrophysical Journal*, 538(2):473–476, 2000. doi: 10.1086/309179.

T. G. F. Li, W. Del Pozzo, S. Vitale, C. Van Den Broeck, M. Agathos, J. Veitch, K. Grover, T. Sidery, R. Sturani, and A. Vecchio. Towards a generic test of the strong field dynamics of general relativity using compact binary coalescence. *Physical Review D*, 85(8):082003, 2012. doi: 10.1103/PhysRevD.85.082003.

LIGO Scientific Collaboration and Virgo Collaboration. Observation of gravitational waves from a binary black hole merger. *Physical Review Letters*, 116(6):1–16, 2016. doi: 10.1103/PhysRevLett.116.061102.

D. V. Lindley. The Future of Statistics: A Bayesian 21st Century. *Advances in Applied Probability*, 7:106–115, 1975. doi: 10.2307/1426315.

LISA Collaboration. Laser Interferometer Space Antenna Simulator. *arXiv preprint arXiv:1702.00786*, 2017.

J. Liu, D. J. Nordman, and W. Q. Meeker. The Number of MCMC Draws Needed to Compute Bayesian Credible Bounds. *The American Statistician*, 06340:1–27, 2016. doi: 10.1080/00031305.2016.1158738.

T. J. Loredo. Bayesian astrostatistics: a backward look to the future. *Astrostatistical Challenges for the New Astronomy*, pages 15–40, 2012. doi: 10.1007/978-1-4614-3508-2_2.

D. J. C. Mackay. Probable networks and plausible predictions - a review of practical Bayesian methods for supervised neural networks. *Network: Computation in Neural Systems*, 6(3):469–505, 1995. doi: 10.1088/0954-898X_6_3_011.

D. J. C. MacKay. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2003. ISBN 9780521642989.

K. Maisinger, M. P. Hobson, and A. N. Lasenby. Maximum-entropy image reconstruction using wavelets. *Monthly Notices of the Royal Astronomical Society*, 347(1):339–354, 2004. doi: 10.1111/j.1365-2966.2004.07216.x.

S. G. Mallat and Z. Zhang. Matching Pursuits With Time-Frequency Dictionaries. *IEEE Transactions on Signal Processing*, 41(12):3397–3415, 1993. doi: 10.1109/78.258082.

S. Martiniani, J. D. Stevenson, D. J. Wales, and D. Frenkel. Superposition enhanced nested sampling. *Physical Review X*, 4(3), 2014. doi: 10.1103/PhysRevX.4.031034.

F. J. Massey. The Kolmogorov-Smirnov Test for Goodness of Fit. *Journal of the American Statistical Association*, 46(253):68– 78, 1951. doi: 10.1080/01621459.1951.10500769.

J. D. McEwen and Y. Wiaux. Compressed sensing for wide-field radio interferometric imaging. *Monthly Notices of the Royal Astronomical Society*, 413(2):1318–1332, 2011. doi: 10.1111/j.1365-2966.2011.18217.x.

N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6):1087–1092, 1953. doi: 10.1063/1.1699114.

P. Mukherjee, D. Parkinson, and A. R. Liddle. A Nested Sampling Algorithm for Cosmological Model Selection. *The Astrophysical Journal*, 638(2):L51–L54, 2006. doi: 10.1086/501068.

I. Murray. *Advances in Markov chain Monte Carlo methods*. PhD thesis, University College London, 2007.

R. Neal. *Bayesian learning for neural networks*. Springer Science & Business Media, 2012. ISBN 1461207452.

R. M. Neal. Slice Sampling. *Annals of statistics*, 31(3):705–741, 2003. doi: 10.1214/aos/1056562461.

Y. Okamoto. Generalized-ensemble algorithms: Enhanced sampling techniques for Monte Carlo and molecular dynamics simulations. In *Journal of Molecular Graphics and Modelling*, volume 22, pages 425–439, 2004. ISBN 1093-3263. doi: 10.1016/j.jmgm.2003.12.009.

J. D. Orazio, A. Loeb, and J. Guillochon. Constraining the Stellar Mass Function from the Deficiency of Tidal Disruption Flares in the Nuclei of Massive Galaxies. *arXiv preprint arXiv:1807.00029*, 2018.

A. Pancoast, B. J. Brewer, T. Treu, D. Park, A. J. Barth, M. C. Bentz, and J. H. Woo. Modelling reverberation mapping data - II. Dynamical modelling of the Lick AGN Monitoring Project 2008 data set. *Monthly Notices of the Royal Astronomical Society*, 445(3):3073–3091, 2014. doi: 10.1093/mnras/stu1419.

D. Parkinson and A. R. Liddle. Bayesian Model Averaging in Astrophysics: A Review. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 6(1):3–14, 2013. doi: 10.1002/sam.11179.

Planck Collaboration. Planck 2013 results. XVI. Cosmological parameters. *Astronomy & Astrophysics*, 571:1–69, 2013. doi: 10.1051/0004-6361/201321591.

Planck Collaboration. Planck 2015. XX. Constraints on inflation. *Astronomy & Astrophysics*, 594:A20, 2016a. doi: 10.1051/0004-6361/201525898.

Planck Collaboration. Planck 2015 results XI. CMB power spectra, likelihoods, and robustness of parameters. *Astronomy and Astrophysics*, 594:A11, 2016b. doi: 10.1051/0004-6361/201526926.

C. Rasmussen. Gaussian processes in machine learning. In *Advanced lectures on machine learning*, pages 63–71. Springer, 2004. doi: 10.1007/978-3-540-28650-9_4.

D. B. Rubin. The Bayesian Bootstrap. *Annals of Statistics*, 9(1):130–134, 1981. doi: 10.1214/aos/1176345338.

R. Salomone, L. F. South, C. C. Drovandi, and D. P. Kroese. Unbiased and Consistent Nested Sampling via Sequential Monte Carlo. *arXiv preprint arXiv:1805.03924*, 2018.

A. Savitzky and M. J. Golay. Smoothing and Differentiation of Data by Simplified Least Squares Procedures. *Analytical Chemistry*, 36(8):1627–1639, 1964. doi: 10.1021/ac60214a047.

F. W. Scholz and M. A. Stephens. K-sample AndersonDarling tests. *Journal of the American Statistical Association*, 82(399):918–924, 1987. doi: 10.1080/01621459.1987.10478517.

F. Sciacchitano, S. Lugaro, and A. Sorrentino. Sparse Bayesian Imaging of Solar Flares. *SIAM Journal on Imaging Sciences*, 12(1):319–343, 2019. doi: 10.1137/18M1204103.

D. W. Scott. *Multivariate Density Estimation: Theory, Practice, and Visualization*. John Wiley & Sons, 2015. ISBN 9781118575536.

S. Sharma. Markov Chain Monte Carlo Methods for Bayesian Data Analysis in Astronomy. *Annual Review of Astronomy and Astrophysics*, pages 213–259, 2017. doi: 10.1146/annurev-astro-082214-122339.

D. Sivia and J. Skilling. *Data analysis: A Bayesian Tutorial*. OUP Oxford, 2006. ISBN 0198568320. doi: 10.2307/1270652.

J. Skilling. Nested Sampling. In *24th International Workshop on Bayesian Inference and Maximum Entropy Methods in Science and Engineering*, volume 735, pages 395–405, 2004. ISBN 0735402175. doi: 10.1063/1.1835238.

J. Skilling. Nested sampling for general Bayesian computation. *Bayesian Analysis*, 1 (4):833–860, 2006. doi: 10.1214/06-BA127.

J. Skilling. Nested sampling's convergence. In *AIP Conference Proceedings*, volume 1193, pages 277–291, 2009. ISBN 9780735407299. doi: 10.1063/1.3275625.

J. Skilling. Bayesian computation in big spaces-nested sampling and Galilean Monte Carlo. In *AIP Conference Proceedings*, volume 1443, pages 145–156, 2012. ISBN 9780735410398. doi: 10.1063/1.3703630.

R. Tibshirani. Regression Selection and Shrinkage via the Lasso. *Journal of the Royal Statistical Society B*, 58(1):267–288, 1996. doi: 10.2307/2346178.

L. Tierney and J. B. Kadane. Accurate Approximations for Posterior Moments and Marginal Densities. *Journal of the American Statistical Association*, 81(393):82–86, 1986.

M. Tipping. Sparse Bayesian Learning and the Relevance Vector Machine. *Journal of Machine Learning Research*, 1:211–244, 2001. doi: 10.1162/15324430152748236.

R. Trotta. Applications of Bayesian model selection to cosmological parameters. *Monthly Notices of the Royal Astronomical Society*, 378(1):72–82, 2007. doi: 10.1111/j.1365-2966.2007.11738.x.

R. Trotta. Bayes in the sky: Bayesian inference and model selection in cosmology. *Contemporary Physics*, 49(2):71–104, 2008. doi: 10.1080/00107510802066753.

J. W. Tukey. Bias and Confidence in Not-Quite Large Samples. *The Annals of Mathematical Statistics*, 29:614, 1958. doi: 10.2307/2237363.

I. Verdinelli and L. Wasserman. Computing Bayes Factors Using a Generalization of thhe Savage-Dickey Density Ratio. *Journal of the American Statistical Association*, 90(430):614–618, 1995.

C. Walter. Point process-based Monte Carlo estimation. *Statistics and Computing*, 27 (1):219–236, 2017. doi: 10.1007/s11222-015-9617-y.

H. P. Warren, J. M. Byers, and N. A. Crump. Sparse Bayesian Inference and the Temperature Structure of the Solar Corona. *The Astrophysical Journal*, 836(2):215, 2017. doi: 10.3847/1538-4357/aa5c34.

R. L. Wasserstein and N. A. Lazar. The ASA's Statement on p-Values: Context, Process, and Purpose. *The American Statistician*, 70(2):129–133, 2016. doi: 10.1080/00031305.2016.1154108.

S. Watanabe. Asymptotic Equivalence of Bayes Cross Validation and Widely Applicable Information Criterion in Singular Learning Theory. *Journal of Machine Learning Research*, 11:3571–3594, 2010.

Y. Wiaux, L. Jacques, G. Puy, A. Scaife, and P. Vandergheynst. Compressed sensing imaging techniques for radio interferometry. *Monthly Notices of the Royal Astronomical Society*, 395(3):1733–1742, 2009. doi: 10.1111/j.1365-2966.2009.14665.x.

N. Wiener. *The interpolation, extrapolation and smoothing of stationary time series.* MIT Press Cambridge, MA, 1949. ISBN 978-0262730051.

D. P. Wipf and B. D. Rao. Sparse Bayesian learning for basis selection. *IEEE Transactions on Signal Processing*, 52(8):2153–2164, 2004. doi: 10.1109/TSP.2004.831016.

D. H. Wolpert and W. G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, 1997. doi: 10.1109/4235.585893.

C. Zucker, E. F. Schlafly, G. M. Green, J. S. Speagle, S. K. N. Portillo, D. P. Finkbeiner, and A. A. Goodman. Mapping Distances across the Perseus Molecular Cloud Using CO Observations, Stellar Photometry, and Gaia DR2 Parallax Measurements. *The Astrophysical Journal*, 869(1):83, 2018. doi: 10.3847/1538-4357/aae97c.