Mon. Not. R. Astron. Soc. **000**, 1–15 (2015) Printed 30 July 2015 (MN LATEX style file v2.2)

PolyChord: next-generation nested sampling

W.J. Handley[1,2⋆], M.P. Hobson[1]† & A.N. Lasenby[1,2]‡
[1] *Astrophysics Group, Cavendish Laboratory, J. J. Thomson Avenue, Cambridge, CB3 0HE, UK*
[2] *Kavli Institute for Cosmology, Madingley Road, Cambridge, CB3 0HA, UK*

Received 30 May 2015

ABSTRACT
PolyChord is a novel nested sampling algorithm tailored for high-dimensional parameter spaces. This paper coincides with the release of PolyChord v1.3, and provides an extensive account of the algorithm. PolyChord utilises slice sampling at each iteration to sample within the hard likelihood constraint of nested sampling. It can identify and evolve separate modes of a posterior semi-independently, and is parallelised using openMPI. It is capable of exploiting a hierarchy of parameter speeds such as those present in CosmoMC and CAMB, and is now in use in the CosmoChord and ModeChord codes. PolyChord is available for download from:
http://ccpforge.cse.rl.ac.uk/gf/project/polychord/

Key words: methods: data analysis — methods: statistical

1 INTRODUCTION

Over the past two decades, Bayesian methods have been increasingly adopted as the standard inference procedure for the rapidly increasing volume of astrophysical data.

Bayesian inference consists of *parameter estimation* and *model comparison*. Parameter estimation is generally performed using Markov-Chain Monte-Carlo (MCMC) methods, such as the Metropolis–Hastings (MH) algorithm and its variants (MacKay 2002). In order to perform model comparison, one must calculate the *evidence*: a high-dimensional integration of the likelihood over the prior density (Sivia & Skilling 2006). MH methods cannot compute this on a usable timescale, hindering the use of Bayesian model comparison in cosmology and astroparticle physics.

A contemporary methodology for computing evidences and posteriors simultaneously is provided by nested sampling (Skilling 2006). This has been successfully implemented in the now widely adopted algorithm MultiNest (Feroz & Hobson 2008; Feroz et al. 2009, 2013). Modern cosmological likelihoods now involve a large number of parameters, with a hierarchy of speeds. MultiNest struggles with high-dimensional parameter spaces, and is unable to take advantage of this separation of speeds. PolyChord aims to address these issues, providing a means to sample high-dimensional spaces across a hierarchy of parameter speeds.

The layout of the paper is as follows: Section 2 is a general overview of parameter estimation and model selection in the context of Bayesian Inference. In Section 3 we describe Skilling's (2006) nested sampling meta-algorithm. We overview the historical implementations of nested sampling in Section 4 and provide an account of Neal's (2000) slice sampling technique. We describe the PolyChord algorithm in detail in Section 5 and demonstrate its efficacy on toy and cosmological problems in Section 6. Section 7 concludes the paper. In addition we provide three appendices. Appendix A describes the procedure for implementing new prior distributions within the context of nested sampling. Appendices B & C describe the mathematics of inferring evidences from the samples produced by nested sampling.

This paper is an extensive overview of our algorithm, which is now in use in several cosmological applications (Planck Collaboration XX 2015). A briefer introduction can be found in Handley et al. (2015).

PolyChord is available for download from the link at the end of the paper.

2 BAYESIAN INFERENCE

In this section, we describe the key concepts of Bayesian inference necessary for understanding the utility of PolyChord. For readers experienced in the field, this section serves to establish nomenclature and notation. For a full discussion of Bayesian inference, we recommend Sivia & Skilling (2006) or part IV of MacKay (2002).

2.1 Nomenclature

Scientific theory is concerned with the construction of predictive models in the context of some dataset $\mathcal{D}$. A typical

⋆ wh260@mrao.cam.ac.uk
† mph@mrao.cam.ac.uk
‡ a.n.lasenby@mrao.cam.ac.uk

model $\mathcal{M}$ contains a set of variable parameters $\theta_{\mathcal{M}}$. One may use $\mathcal{M}$ to calculate the probability of observing the data given a specific parameter choice:

$$\mathrm{P}(\mathcal{D}|\theta_{\mathcal{M}}, \mathcal{M}) \equiv \mathcal{L}. \tag{1}$$

This distribution on $\mathcal{D}$ is termed the *likelihood* $\mathcal{L}$. From a Bayesian standpoint a model must also specify our initial degree of belief on the parameters $\theta_{\mathcal{M}}$:

$$\mathrm{P}(\theta_{\mathcal{M}}|\mathcal{M}) \equiv \pi, \tag{2}$$

This distribution on $\theta_{\mathcal{M}}$ is termed the *prior* π. Typically this is a parametric distribution which quantifies our initial assumptions on the scale and spread of the parameters[1].

The likelihood (1) is conditioned on a set of chosen values for the model parameters $\theta_{\mathcal{M}}$. One may marginalise out the dependence on $\theta_{\mathcal{M}}$ by integrating over the prior distribution:

$$\mathrm{P}(\mathcal{D}|\mathcal{M}) \equiv \mathcal{Z} = \int \mathrm{P}(\mathcal{D}|\theta_{\mathcal{M}}, \mathcal{M})\mathrm{P}(\theta_{\mathcal{M}}|\mathcal{M})\, d\theta_{\mathcal{M}}. \tag{3}$$

This quantity is termed the *evidence* $\mathcal{Z}$, or *marginalised likelihood*, and gives the probability of observing the data $\mathcal{D}$, conditioned on the model $\mathcal{M}$. Suppressing explicit dependence on the model, the evidence computation can be written as:

$$\mathcal{Z} = \int \mathcal{L}(\theta)\pi(\theta)\, d\theta. \tag{4}$$

2.2 Parameter estimation

If the prior has been specified, Bayes theorem allows us to invert the conditioning in equation (1) and find the *posterior* $\mathcal{P}$ by combining the likelihood, prior and evidence:

$$\mathrm{P}(\theta_{\mathcal{M}}|\mathcal{D}, \mathcal{M}) = \frac{\mathrm{P}(\mathcal{D}|\theta_{\mathcal{M}}, \mathcal{M})\mathrm{P}(\theta_{\mathcal{M}}|\mathcal{M})}{\mathrm{P}(\mathcal{D}|\mathcal{M})}, \tag{5}$$

which is schematically written as:

$$\mathcal{P} = \frac{\mathcal{L} \times \pi}{\mathcal{Z}}. \tag{6}$$

This describes how our initial knowledge π of the parameters updates to $\mathcal{P}$ in light of the data $\mathcal{D}$. Calculation of the posterior $\mathcal{P}(\theta)$ is the domain of *parameter estimation*, and in high dimensions is best performed by sampling the space with a Markov-Chain Monte-Carlo approach (MCMC). Examples include Metropolis–Hastings, Gibbs sampling and Slice sampling. For the most part, the evidence $\mathcal{Z}$ is ignored during such calculations, and one works with an unnormalised posterior $\mathcal{P} \propto \mathcal{L} \times \pi$.

2.3 Model comparison

Of equal importance in scientific investigation is *model comparison*. Typically one has multiple competing models

[1] Common examples include a uniform distribution between two bounds, or a Gaussian distribution with specified mean and variance.

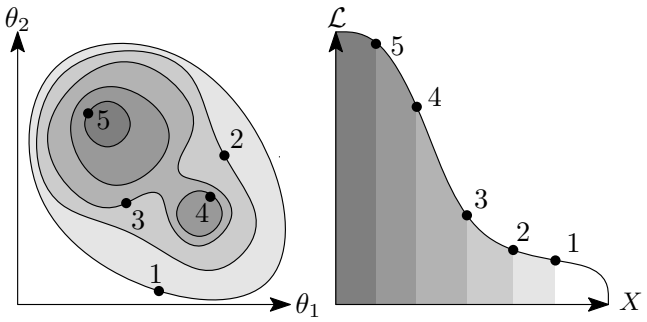

Figure 1. The nested sampling volume transformation. Left: five iso-likelihood contours of a two-dimensional multi-modal likelihood function $\mathcal{L}(\theta)$. Each contour encloses some fraction of the prior X, indicated by colour. Right: Likelihood $\mathcal{L}$ as a function of the volume X enclosed by the contour. The evidence is the area under this curve.

$\{\mathcal{M}_1, \mathcal{M}_2, \cdots\}$, each with their own parameters and assumptions. The data $\mathcal{D}$ are able to decide on the relative merits of each of these models via Bayes theorem:

$$\mathrm{P}(\mathcal{M}_i|\mathcal{D}) = \frac{\mathrm{P}(\mathcal{D}|\mathcal{M}_i)\mathrm{P}(\mathcal{M}_i)}{\mathrm{P}(\mathcal{D})}, \tag{7}$$

$$= \frac{\mathcal{Z}_i\pi_i}{\sum_j \mathcal{Z}_j\pi_j}. \tag{8}$$

In contrast to parameter estimation, the evidences of each model $\mathcal{Z}_i$ take the leading role in model comparison. One typically will choose uniform priors on the models, $\pi_i \equiv \mathrm{P}(\mathcal{M}_i) = \text{const}$, and then choose to use the model with the highest evidence. However, when evidences are similar in magnitude, the correct Bayesian approach is to make inferences by marginalising over all models considered. If there is a common derived parameter y, with marginalised posterior $\mathrm{P}(y|\mathcal{D}, \mathcal{M}_i)$ then one may produce the fully marginalised posterior:

$$\mathrm{P}(y|\mathcal{D}) = \frac{\sum_i \mathrm{P}(y|\mathcal{D}, \mathcal{M}_i)\mathcal{Z}_i\pi_i}{\sum_j \mathcal{Z}_j\pi_j}. \tag{9}$$

This fully Bayesian approach has been historically under utilised due to the difficulties in computing the evidence numerically from the integral (4).

3 NESTED SAMPLING

PolyChord falls into a category of sampling algorithms known as *nested sampling*. In order to explain the advances that PolyChord has made, it is first necessary to describe the nested sampling meta-algorithm. Readers familiar with the theory may skip to Section 4.

Computing the evidence (4) typically involves an integral over a high-dimensional parameter space, only a small fraction of which contributes to $\mathcal{Z}$. The size and position of the region surrounding the peak(s) will not be known *a priori*, and in high dimensions is hard to find (see Figure 1).

Algorithms need to be able quickly to compress the parameter space from the prior onto the posterior. In order to perform parameter estimation it needs to produce samples from the posterior, and to perform model comparison

it should be able to calculate the evidence. Nested sampling (Skilling 2006) offers a means of doing all of these tasks simultaneously.

3.1 Compressing the space

Nested sampling maintains a population of $n_{\rm live}$ *live points* within a region of the parameter space. These points are sequentially updated so that the region that they occupy contracts around the peak(s) of the posterior.

One begins by sampling $n_{\rm live}$ points from the prior distribution $\pi(\theta)$. At iteration i, the point with the lowest likelihood $\mathcal{L}_i$ is deleted, and then replaced by a new point. The new point is drawn from the prior, subject to the constraint that its likelihood is greater than $\mathcal{L}_i$.

The fraction of the prior contained within an iso-likelihood contour $\mathcal{L}(\theta) = \mathcal{L}$ is denoted the *prior volume*:

$$X(\mathcal{L}) = \int_{\mathcal{L}(\theta)>\mathcal{L}} \pi(\theta) d\theta. \tag{10}$$

Since the live points are always drawn uniformly from $\pi(\theta)$, at iteration i the volume containing the live points will contract on average by a factor of $n_{\rm live}/(n_{\rm live}+1)$. Initially the prior volume is 1, so at iteration i:

$$\langle X_i \rangle = \left(\frac{n_{\rm live}}{n_{\rm live}+1}\right)^i \approx e^{-i/n_{\rm live}}. \tag{11}$$

The live points thus compress the prior *exponentially*. As the nested sampling run progresses, one is left with a sequence of discarded points (termed *dead points*). Each dead point will have a set of parameter values θ_i, a likelihood $\mathcal{L}_i$ and an estimated prior volume X_i.

3.2 Evidence estimation

We can use the dead and live points to estimate the evidence. By differentiating the prior volume (10), we may re-write the evidence calculation (4) as an integral over a single variable:

$$\mathcal{Z} = \int_0^1 \mathcal{L}(X) dX. \tag{12}$$

This is detailed graphically in Figure 1. We may thus estimate the evidence by quadrature:

$$\mathcal{Z} \approx \sum_{i\in \rm dead} w_i \mathcal{L}_i, \tag{13}$$

where for simplicity we take $w_i = X_{i-1} - X_i$. Of course, this is only an estimate, since we are inferring the mean values $\langle X_i \rangle$ from the sampling procedure. One may however estimate the error in our inference, the full details of which can be found in Appendix B.

3.3 Parameter estimation

Nested sampling can also perform parameter estimation by using the dead and live points as samples from the posterior, provided that the ith point is given the importance weighting:

$$p_i = \frac{w_i \mathcal{L}_i}{\mathcal{Z}}, \tag{14}$$

where w_i is the prior volume of the shell in which point i was sampled.

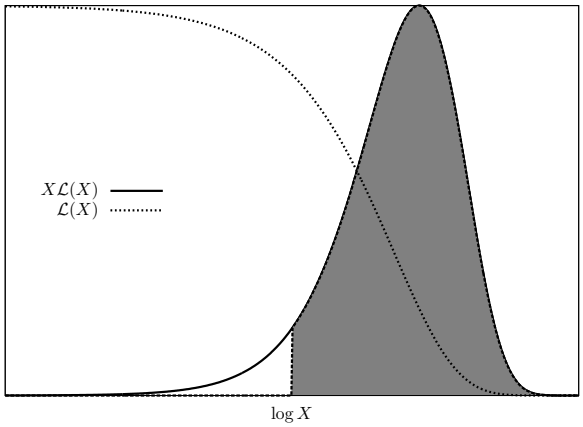

Figure 2. Plot of a generic likelihood as a function of the prior volume $\mathcal{L}(X)$. In high dimensions, the likelihood is only visible if plotted against $\log X$ (dashed curve). However, the evidence is better visualised by plotting $X\log(X)$ (solid curve). The area under the solid curve corresponds to the evidence. The magnitude of the solid curve is proportional to the importance weighting. Nested sampling proceeds from high to low volumes. After some time, the live points no longer contribute significantly to the evidence, and the algorithm terminates at this point.

3.4 Algorithm termination

As nested sampling proceeds, the likelihoods $\mathcal{L}_i$ monotonically increase, but the weights w_i monotonically decrease. This results in a peak in importance weights (14) that can be seen in Figure 2. We terminate the algorithm once the remaining posterior mass (white region) left in the live points is some small fraction of the currently calculated evidence (dark region). The posterior mass left in the live points at iteration i can be estimated by:

$$\mathcal{Z}_{\rm live} \approx \langle \mathcal{L} \rangle_{\rm live} X_i, \tag{15}$$

where the average is taken over the live points. Since this is typically an underestimate at early times, this will not cause premature termination.

3.5 The unit hypercube

Each iteration of nested sampling requires one to sample from the prior (subject to a hard likelihood constraint). Typically, priors are defined in terms of simple analytic functions such as uniform or Gaussian distributions, and may be sampled using inverse transform sampling.

In the one-dimensional case, this amounts to converting a uniform random variable (which are easy to generate) into a variable sampled from a general distribution $f(\theta)$. One first finds its cumulative distribution function (CDF):

$$F(\theta) = \int_{-\infty}^{\theta} f(\theta') d\theta', \tag{16}$$

computes the inverse of the CDF, and then applies this function to a uniform random variable $x \sim U(0,1)$ to generate a variable $\theta = F^{-1}(x)$, which is distributed according to $f(\theta)$.

In the general D-dimensional case, one calculates D

conditional distributions $\{f_i : i = 1\ldots, D\}$: by marginalising over parameters with indices greater than i and conditioning on parameters with indices less than i:

$$f_i(\theta_i|\theta_{i-1},\ldots,\theta_1) = \frac{\int f_i(\theta) d\theta_{i+1}\ldots d\theta_N}{\int f_i(\theta) d\theta_i \ldots d\theta_N}, \tag{17}$$

Integrating these yields D conditional CDFs:

$$x_i = F_i(\theta_i|\theta_{i-1},\ldots,\theta_1) = \int_0^{\theta_i} f_i(\theta_i'|\theta_{i-1},\ldots,\theta_1) d\theta_i'. \tag{18}$$

Inverting this gives $\theta_i = F_i^{-1}(x_i|\theta_{i-1},\ldots,\theta_1)$, which constitutes a set of relations sequentially transforming D uniform random variables $\{x_i\}$ into $\{\theta_i\}$ distributed according to $f(\theta)$.

In many cases, the prior $\pi(\theta)$ is separable, and the above equations are easily calculated. For sections of the parameters which are not separable, the calculation can become more involved. We include a few demonstrations of this procedure in Appendix A.

Nested sampling can thus be performed in the unit D-dimensional hypercube, $\mathbf{x} \in [0,1]^D$, defining a new likelihood function via $\mathcal{L}(\theta) = \mathcal{L}(\mathbf{F}^{-1}(\mathbf{x}))$. This has numerous advantages, the first being that one only needs to be able to generate uniform random variables in $[0,1]$. The second is more subtle; it is more natural to define a distance metric in the unit hypercube than in the physical space. Unit hypercube variables all have the same dimensionality: probability.

4 SAMPLING WITHIN AN ISO-LIKELIHOOD CONTOUR

Now that the nested sampling meta-algorithm has been described, we briefly review the various instantiations that exist, and introduce PolyChord as an algorithm utilising slice sampling at each iteration to generate new live points.

The most challenging aspect of nested sampling is drawing a new point from the prior subject to the hard likelihood constraint $\mathcal{L} > \mathcal{L}_i$. This may be done in a variety of ways, and distinguishes the various historical implementations.

4.1 Previous Methods

For some problems, the iso-likelihood contour is known analytically, allowing one to construct a sampling procedure specific to that problem. This is demonstrated by Keeton (2011), and can be useful for testing nested sampling's theoretical behaviour. In most cases, however, the likelihood contour is unknown a-priori, so a more numerical approach must be taken.

Mukherjee et al. (2006) implemented a rejection sampling method, which was later incorporated into the widely-used MultiNest algorithm (Feroz & Hobson 2008; Feroz et al. 2009, 2013). These algorithms sample by using the live points to construct a set of intersecting ellipsoids which together aim to enclose the likelihood contour, and then performs rejection sampling within the ellipsoids. Whilst being an excellent algorithm for modest numbers of parameters, any rejection sampling algorithm has an exponential scaling with dimensionality that eventually emerges.

An alternative approach (the one initially envisaged by Skilling) is to sample with the hard likelihood constraint using a Markov-Chain based procedure. One makes several steps according to some proposal distribution until one is satisfied an independent sample is produced. This has significant advantages over a rejection-based approach, the most obvious being that the scaling with dimensionality is polynomial rather than exponential. In rejection sampling, points are drawn until one is found within the likelihood contour (often with extremely low efficiency). Using a Markov-chain approach however, (correlated) points are continually generated within the contour, until one is happy that a sample independent from the initial seed has been generated. These "intra-chain points" which we term *phantom points* have the potential to provide a great deal more information.

A traditional Metropolis–Hastings (MH) or Gibbs sampling approach may be utilised, but in general such algorithms are ill-suited to sampling from a hard likelihood constraint without a significant amount of tuning of a proposal matrix. This is examined in section 6 of Feroz & Hobson (2008).

Galilean (Hamiltonian) sampling (Feroz & Skilling 2013; Betancourt 2011) improves upon the traditional MH sampler by using proposal points generated by reflecting off iso-likelihood contours. This however requires gradients to be calculated, and can become inefficient if the step size is chosen incorrectly, or if the contour has a shape which is difficult to 'step back into'

Diffusive nested sampling (Brewer et al. 2009) is an alternative and promising variation on Skilling's (2006) algorithm, which utilises MCMC to explore a mixture of nested probability distributions. Since it is MCMC based, it scales well with dimensionality. In addition, it can deal with multimodal and degenerate posteriors, unlike traditional MCMC. It does however have multiple tuning parameters.

4.2 Slice sampling

We have found that a Markov-Chain based procedure utilising Neal's (2000) slice sampling at each step is well suited to sampling uniformly within an iso-likelihood contour. Radford Neal initially proposed slice sampling as an effective methodology for generating samples numerically from a given posterior $\mathcal{P}(\theta)$. One first chooses a 'slice' (or probability level) $\mathcal{P}_0$ uniformly within $[0, \mathcal{P}_{\max}]$. One then samples uniformly within the θ-region defined by $\mathcal{P}(\theta) > \mathcal{P}_0$. The similarity with the iso-likelihood contour sampling required by nested sampling should be clear. In the one-dimensional case, he suggests the sampling procedure detailed in Figure 3.

This procedure for sampling within a likelihood bound is ideal for nested sampling. It samples uniformly with minimal information: an initial bound size w, and a point x_0 that is within the contour. In general w must be chosen so that it is roughly the size of the bound, but if one overestimates it then the bounds will contract exponentially. Indeed, one may consider this as being equivalent to a prior space compression (11) with $n_{\text{live}} = n_{\text{dims}} = 1$. As a starting point, one may use one of the live points, which is already uniformly sampled. Since the procedure above satisfies detailed balance, this will produce a point which is also uniformly sampled within the iso-likelihood contour.

In higher dimensions, Neal (2000) suggests a variety of

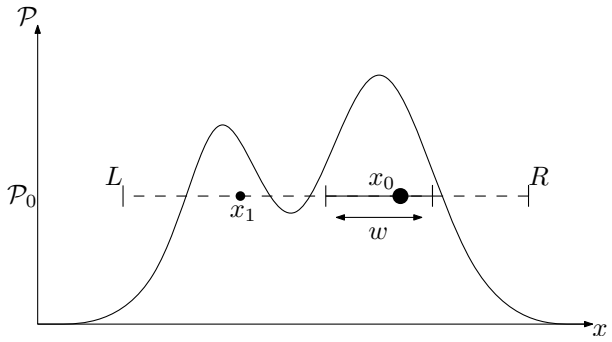

Figure 3. Slice sampling in one dimension. Given a probability level (or slice) $\mathcal{P}_0$, slice sampling samples within the horizontal region defined by $\mathcal{P} > \mathcal{P}_0$. From an initial point x_0 within the slice ($\mathcal{P}(x_0) > \mathcal{P}_0$), a new point x_1 is generated within the slice with a distribution $P(x_1|x_0)$. External bounds are first set on the slice $\hat{L} < x_0 < \hat{R}$ by uniformly expanding a random initial bound of width w until they lie outside the slice (Neal terms this the *stepping out* procedure). x_1 is then sampled uniformly within these bounds. If x_1 is not in the slice, then $\hat{L}$ or $\hat{R}$ is replaced with x_1, ensuring that x_0 is still within the slice. This procedure is guaranteed to generate a new point x_1, and satisfies detailed balance $P(x_0|x_1) = P(x_1|x_0)$. Thus, if x_0 is drawn from a uniform distribution within the slice, so is x_1.

MCMC-like methods. The simplest of these is implemented by sampling each of the parameter directions in turn. Since each one-dimensional slice requires $\sim \mathcal{O}$(a few) likelihood calculations, the number of likelihood calculations required scales linearly with dimensionality, providing the region is efficiently navigated. Multi-dimensional slice sampling has many of the benefits of a traditional MH approach, and uses a proposal distribution which is much more efficient at sampling a hard likelihood constraint.

Aitken & Akman (2013) have already applied this procedure to nested sampling. This works exceptionally well for cases in which the parameters are non-degenerate. However, this becomes inefficient in the case of correlated parameters, or curving degeneracies.

5 THE PolyChord ALGORITHM

PolyChord implements several novel features compared to Aitken & Akman's (2013) slice-based nested sampling. It utilises slice sampling in a manner that uses the information present in the live and phantom points to deal with correlated posteriors. PolyChord also uses a general clustering algorithm that identifies and evolves separate modes of the posterior semi-independently, and infers local evidence values. In addition, it has the option of implementing fast-slow parameters, which is extremely effective in its combination with CosmoMC (Lewis & Bridle 2002). This is termed CosmoChord, which may be downloaded from the link at the end of the paper.

The algorithm is written in FORTRAN95 and parallelised using openMPI. It is optimised for the case where the dominant cost is the generation of a new live point. This is frequently the case in astrophysical applications, either due to high dimensionality, or to costly likelihood evaluation.

5.1 Multi-dimensional slice sampling

At each iteration i of nested sampling, we generate a new randomly sampled point within the iso-likelihood contour $\mathcal{L}_i$ by our variant of D-dimensional slice sampling. Slice sampling is performed in the unit hypercube with hypercube coordinates denoted in bold ($\mathbf{x}$).

At each iteration i of the nested sampling algorithm, one of the live points is chosen at random as a start point for a new chain with hypercube coordinate $\mathbf{x}_0$. We then make a one-dimensional slice sampling step (Figure 3) with initial width w in a random direction $\hat{\mathbf{n}}_0$ chosen from a probability distribution $\mathrm{P}(\hat{\mathbf{n}})$. This generates a new point $\mathbf{x}_1$ which is uniformly sampled in the unit hypercube, but is correlated to $\mathbf{x}_0$. This process is repeated $n_{\rm repeats}$ times, with $\mathbf{x}_{j-1}$ forming the start point for a slice along $\hat{\mathbf{n}}_{j-1}$ to produce $\mathbf{x}_j$. This procedure is illustrated in the right hand half of Figure 4.

Since the probability of drawing $\mathbf{x}_j$ from $\mathbf{x}_{j-1}$ is the same as the probability of drawing $\mathbf{x}_{j-1}$ from $\mathbf{x}_j$, this procedure satisfies detailed balance. Thus, the resulting chain will ergodically be uniformly distributed within the iso-likelihood contour. This also applies to multi-modal posteriors, with the chance of jumping out a mode being equal to the chance of jumping back in.

The length of the chain $n_{\rm repeats}$ should be large enough so that the final point of the chain is decorrelated from the start point. This final point may now be considered to be a new uniformly sampled point from the prior distribution subject to the hard likelihood constraint. The intermediate points are saved and stored as phantom points. Whilst phantom points are correlated, they are useful in providing additional information and posterior points.

There are several elements of this which are left undetermined, namely the probability distribution $\mathrm{P}(\hat{\mathbf{n}})$, the initial width w, and the chain length $n_{\rm repeats}$. These issues are addressed in the next section.

5.2 Contour whitening

In order to determine an optimal $\mathrm{P}(\hat{\mathbf{n}})$ and w, an algorithm will need some knowledge of the contour in which the chain is progressing. This information can be supplied by the set of live and phantom points which are already uniformly distributed within the contour. We use the sample covariance matrix of the live and phantom points as a proxy for the size and shape of the contour.

Uniformly sampled points remain uniformly sampled under an affine transformation. The covariance matrix is used to construct an affine transformation which "whitens" the contour. Sampling is then performed in this whitened space, which we term the *sampling space*. In the sampling space, the contour has size $\sim \mathcal{O}(1)$ in every direction. This means that one may choose the initial step size as $w = 1$.

To transform from $\mathbf{x}$ in the unit hypercube to $\mathbf{y}$ in the sampling space we use the relation:

$$\mathrm{L}^{-1}\mathbf{x} = \mathbf{y}, \tag{19}$$

where L is the Cholesky decomposition of the covariance matrix $\Sigma = \mathrm{L}\mathrm{L}^T$. This is illustrated further in Figure 4.

Working in the sampling space our choice of $\mathrm{P}(\hat{\mathbf{n}})$ is inspired by the default choice of CosmoMC (Lewis 2013).

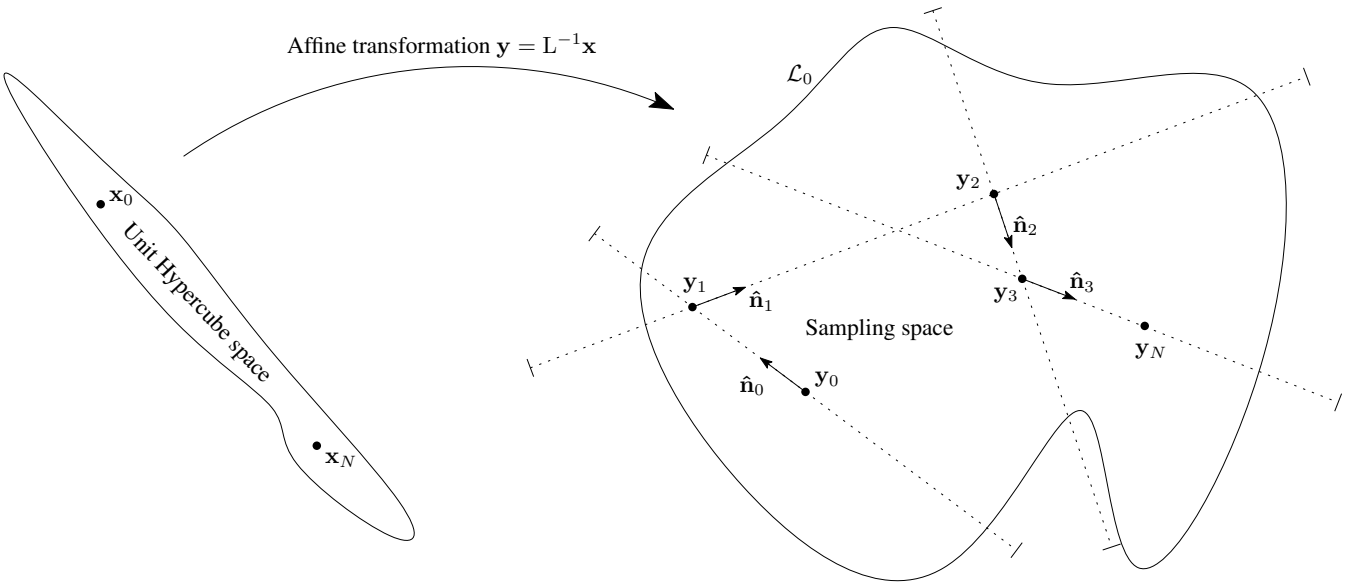

Figure 4. Slice sampling in D dimensions. We begin by "whitening" the unit hypercube by making a linear transformation which turns a degenerate contour into one with dimensions $\sim \mathcal{O}(1)$ in all directions. This is a linear skew transformation defined by the inverse of the Cholesky decomposition of the live points' covariance matrix. We term this whitened space the *sampling space*. Starting from a randomly chosen live point $\mathbf{x}_0$, we pick a random direction and perform one-dimensional slice sampling in that direction (Figure 3), using $w = 1$ in the sampling space. This generates a new point $\mathbf{x}_1$ in $\sim \mathcal{O}$(a few) likelihood evaluations. This process is repeated $\sim \mathcal{O}(n_{\rm dims})$ times to generate a new uniformly sampled point $\mathbf{x}_N$ which is decorrelated from $\mathbf{x}_0$.

Here, a randomly oriented orthonormal basis is chosen, and these directions are chosen in a random order. Once a basis is exhausted, a new basis is chosen. This approach satisfies detailed balance, and mixes rapidly.

The choice of $n_{\rm repeats}$ is slightly harder to justify. We find that for distributions with roughly convex contours $n_{\rm repeats} \sim \mathcal{O}(n_{\rm dims})$ is sufficient, with the constant of proportionality being 2—6. For more complicated contour shapes, one may require much larger values of $n_{\rm repeats}$.

This procedure has the advantage of being dynamically adaptive, and requires no tuning parameters. However, this "whitening" process is ineffective for pronounced curving degeneracies. This will be discussed in detail in Section 6.4.

5.3 Clustering

Multi-modal posteriors are a challenging problem for any sampling algorithm. "Perfect" nested sampling (i.e. the entire prior volume enclosed by the iso-likelihood contour is sampled uniformly) in theory solves multi-modal problems as easily as uni-modal ones. In practice however, there are two issues.

First, one is limited by the resolution of the live points. If a given mode is not populated by enough live points, it runs the risk of "dying out". Indeed, a mode may be entirely missed if the density of live points is too low. In many cases, this problem can be alleviated by increasing the number of live points.

Second, and more importantly for PolyChord, the sampling procedure may not be appropriate for multi-modal problems. We "whiten" the unit hypercube using the covariance matrix of live points. For far-separated modes, the covariance matrix will not approximate the dimensions of the contours, but instead falsely indicate a high degree of correlation. It is therefore essential for our purposes to have PolyChord recognise and treat modes appropriately.

This methodology splits into two distinct parts: (i) recognising that clusters are there, and (ii) evolving the clusters semi-independently.

5.3.1 Cluster recognition

Any cluster recognition algorithm can be substituted at this point. One must take care that this is not run too often, or one runs the risk of adding a large overhead to the calculation. In practice, checking for clustering every $\sim \mathcal{O}(n_{\rm live})$ iterations is sufficient, since the prior will have only compressed by a factor e. We encourage users of PolyChord to experiment with their own preferred cluster recognition, in addition to that provided and described below.

It should be noted that the live points of nested sampling are amenable to most cluster recognition algorithms for two reasons. First, all clusters should have the same density of live points in the unit hypercube. Second, there is no noise (i.e. outside of the likelihood contour there will be no live points). Many clustering algorithms struggle when either of these two conditions is not satisfied.

We therefore choose a relatively simple variant of the k-nearest neighbours algorithm to perform cluster recognition. If two points are within one another's k-nearest neighbours, then these two points belong to the same cluster. We iterate k from 2 upwards until the clustering becomes stable (the cluster decomposition does not change from one k to the next). If sub-clusters are identified, then this process is repeated on the new sub-clusters.

5.3.2 Cluster evolution

An important novel feature comes from what one does once clusters are identified.

First, when spawning from an existing live point, the whitening procedure is now defined by the covariance matrix of the live points within that cluster. This solves the issue detailed above.

Second, by choosing a random initial live point as a seed, PolyChord would naively spawn live points into a mode with a probability proportional to the number of live points in that mode. In fact, what it should be doing is to spawn in proportion to the volume fraction of that mode. In general, these will be approximately the same, but numerical experiments show that the difference between these two ratios exhibits random-walk like behaviour, leading to biases in evidence calculations, or worse, cluster death.

Instead, we keep track of an estimate of the volume in the same manner as equation (11), and choose the mode to spawn into in proportion to that estimate. Further, one may track the errors in this estimate, which contribute to the overall evidence error. This methodology is documented fully in Appendix C.

Thus, the point to be killed off is still the global lowest-likelihood point, but we control the spawning of the new live point into clusters by using our estimates of the volumes of each cluster. We call this 'semi-independent', because it retains global information, whilst still treating the clusters as separate entities.

When spawning within a cluster, we determine the cluster assignment of the new point by which cluster it is nearest to. It does not matter if clusters are identified too soon; the evidence calculation will remain consistent.

In addition to keeping track of local volumes, we may keep track of local evidences. At the moment of splitting, the existing evidence in the initial cluster is partitioned between the new sub-clusters. Upon algorithm completion, one is left with an estimate of the proportion of the evidence contained within each cluster, and thus a measure of the importance of the various modes. By partitioning the local evidences at cluster recognition, the local evidences will sum to give the total evidences, to within the error on our inference.

5.4 Parallelisation

PolyChord is parallelised by openMPI using a master-slave structure. One master process takes the job of organising all of the live points, whilst the remaining $n_{\rm procs}-1$ "slave" processes take the job of finding new live points. This layout is optimised for the case where the dominant cost is the generation of a new live point due to the calculation of relatively expensive likelihoods.

When a new live point is required, the master process sends a random live point and the Cholesky decomposition to a waiting slave. The slave then, after some work, signals to the master that it is ready and returns a new live point and the intra-chain points to the master.

A point generated from an iso-likelihood contour $\mathcal{L}_i$ is usable as a new live point for an iso-likelihood contour $\mathcal{L}_j > \mathcal{L}_i$, providing it is within both contours. One may keep slaves continuously active, and discard any points returned which are not usable. The probability of discarding a point is proportional to the volume ratio of the two contours, so if too many slaves are used, then most will be discarded. The parallelisation goes as:

$$\text{Speedup}(n_{\rm procs}) = n_{\rm live}\log\left[1+\frac{n_{\rm procs}}{n_{\rm live}}\right], \qquad (20)$$

and is illustrated in Figure 5. As a rule, PolyChord parallelises well for $n_{\rm procs} < n_{\rm live}$, but from exhibits a law of diminishing returns. In practice, $n_{\rm procs} = n_{\rm live}/5$ yields $\sim 90\%$ parallelisation efficiency, and since the number of live points is typically ~ 500, this is more than sufficient for currently available openMPI architectures, and certainly superior to the parallelisation of the standard Metropolis–Hastings algorithm.

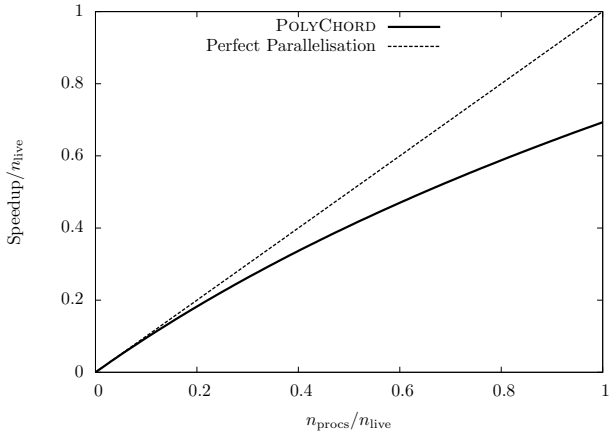

Figure 5. Parallelisation of PolyChord. The algorithm parallelises nearly linearly, providing that $n_{\rm procs} < n_{\rm live}$. For most astronomical applications this is more than sufficient.

5.5 Posterior bulking

In addition to lending information on the scale and shape of a contour, phantom points can also be used as posterior samples. Correlations between samples are unimportant for the purposes of parameter estimation, providing one has enough to be well mixed. We may thus use the importance weighting detailed in (14) with w_i being set to the volume of the live-point shell which they occupy.

For high-dimensional cosmological applications, this results in a very large number ($\gg$GB) of posterior samples being produced, so PolyChord thins these samples. From a user's perspective, one supplies a parameter which determines the fraction of phantom points to keep.

5.6 Fast-slow parameters and CosmoChord

In cosmological applications, likelihoods can exhibit a hierarchy of parameters in terms of calculation speed (Lewis 2013). Consequently, a likelihood may be quickly recalculated if one changes only a certain subset of the parameters. For PolyChord it is very easy to exploit such a hierarchy. Our transformation to the sampling space is laid out so that if parameters are ordered from slow to fast, then this hierarchy is automatically exploited: a Cholesky decomposition,

being a upper-triangular skew transformation, mixes each parameter only with faster parameters.

From a user's perspective, PolyChord does this reordering in the hypercube automatically when provided with details of the hierarchy.

Further to this, one may use the fast directions to extend the chain length by many orders of magnitude. This helps to ensure an even mixing of live points. PolyChord automatically times likelihood calculation speeds, so the user just has to provide what fraction of time PolyChord should be spending on each subset of the parameters, and the algorithm will oversample accordingly.

5.7 Tuning parameters

From a user's perspective, the PolyChord algorithm has two tuning paramaters: $n_{\rm live}$ and $n_{\rm repeats}$, which are detailed below.

The authors believe that these tuning parameters are fairly straightforward to set in comparison to existing algorithms. More importantly, the number of tuning parameters does not scale with the dimensionality of the problem. This is in contrast to Metropolis–Hastings and Gibbs sampling, which require a proposal matrix to be supplied[2].

There are also several other options controlling run time behaviour, such as the production of equally weighted posterior samples, whether or not to perform clustering and the production and use of files allowing PolyChord to resume from a previous run. These are documented in the input files supplied with the code.

Resolution $n_{\rm live}$

This is a generic nested sampling parameter. $n_{\rm live}$ indicates the number of live points maintained throughout the algorithm. Increasing $n_{\rm live}$ causes nested sampling to contract more slowly in volume (equation 11), and consequently sample the space more thoroughly. Thus, it can be thought of as a resolution parameter. Run time scales $\sim \mathcal{O}(n_{\rm live})$

If set too low, posterior modes may be missed. Increasing $n_{\rm live}$ increases the accuracy of the inference of $\mathcal{Z}$, since the evidence error scales $\sim \mathcal{O}\left(n_{\rm live}^{-1/2}\right)$.

Reliability $n_{\rm repeats}$

This is a PolyChord specific parameter. It corresponds to the length of the slice sampling chain used to generate a new live point. Increasing this parameter decreases the correlation between live points, and hence increases the reliability of the evidence inference. Posterior estimations, however, remain accurate even in the event of low $n_{\rm repeats}$.

Setting this too low can result in correlation between live points, and unreliable evidence estimates. Typically, setting this $\sim \mathcal{O}(3 \times n_{\rm dims})$ is sufficient, but for curving degeneracies one may need significantly longer chains. Run time scales $\sim \mathcal{O}(n_{\rm repeats})$.

[2] Proposal matrices may be learnt during run-time. However, this learning step can take some time and may reduce the efficacy of these approaches.

The total number of live and phantom points $n_{\rm live} \times n_{\rm repeats}$ should be large enough that reliable covariance matrices can be calculated. Other than this, the two tuning parameters have independent effects on the algorithm.

In general, $n_{\rm repeats}$ should be scaled linearly with dimensionality D, since one must decorrelate in D independent directions. For typical likelihoods, the logarithmic volume compression from prior to posterior will scale as D. Finally, to keep evidence estimation error constant, the number of live points must be scaled with D. These three effects together mean that PolyChord has a theoretical run time scaling $\sim \mathcal{O}(D^3)$.

6 PolyChord IN ACTION

We aim to showcase PolyChord as both a high-dimensional evidence calculator, and multi-modal posterior sampler. We begin by comparing its dimensionality scaling with MultiNest. We then demonstrate its clustering capabilities in high dimensions, and on difficult clustering problems. PolyChord is shown to perform well on moderately pronounced curving degeneracies, and its implementation in CosmoMC is discussed.

6.1 High-dimensional evidences

As an example of the strength of PolyChord as a high-dimensional evidence estimator, we compare it to MultiNest on a Gaussian likelihood in D dimensions. In both cases, convergence is defined as when the posterior mass contained in the live points is 10^{-2} of the total calculated evidence. We set $n_{\rm live} = 25D$, so that the evidence error remains constant with D. MultiNest was run in its default mode with importance nested sampling and expansion factor $e = 0.1$. Whilst constant efficiency mode has the potential to reduce the number of MultiNest evaluations, the low efficiencies required in order to generate accurate evidences negate this effect.

With these settings, PolyChord produces consistent evidence and error estimates with an error ~ 0.4 log units (Figure 6). Using importance nested sampling, MultiNest produces estimates that are within this accuracy.

Figure 7 shows the number of likelihood evaluations $N_{\mathcal{L}}$ required to achieve convergence as a function of dimensionality D. Even on a simple likelihood such as this, PolyChord shows a significant improvement over MultiNest in scaling with dimensionality. PolyChord at worst scales as $N_{\mathcal{L}} \sim \mathcal{O}(D^3)$, whereas MultiNest has an exponential scaling which emerges in higher dimensions. However, we must point out that a good rejection algorithm like MultiNest will always win in low dimensions. We therefore recommend using MultiNest for low dimensional problems, although it should be noted that MultiNest's clustering is ineffective in modest dimensionalities.

6.2 Clustering and local evidences

To demonstrate PolyChord's clustering capability we report its performance on a "Twin Peaks" and Rastrigin likelihood.

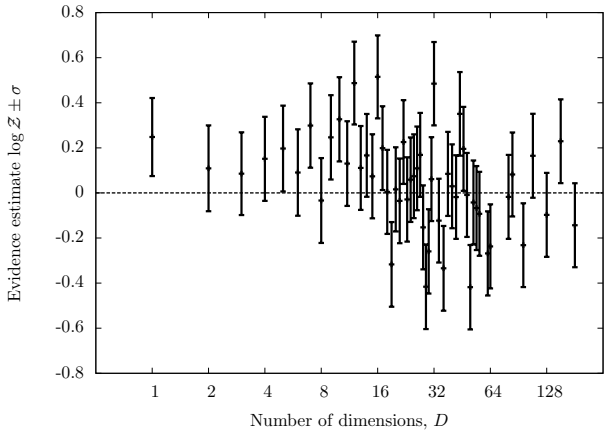

Figure 6. Evidence estimates and errors produced by PolyChord for a Gaussian likelihood as a function of dimensionality. The dashed line indicates the correct analytic evidence value.

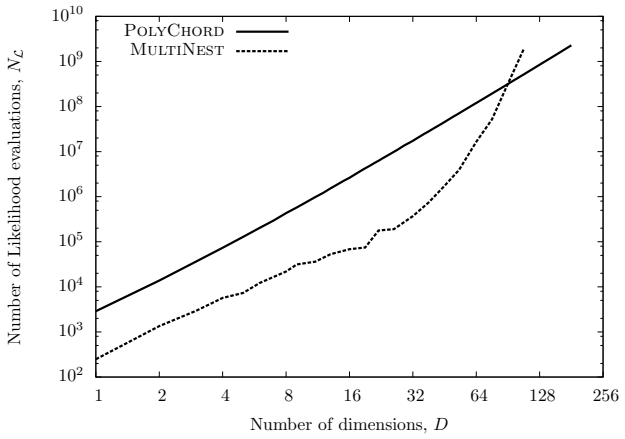

Figure 7. Comparing PolyChord with MultiNest using a Gaussian likelihood for different dimensionalities. PolyChord has at worst $N_{\mathcal{L}} \sim \mathcal{O}(D^3)$, whereas MultiNest has an exponential scaling that emerges at high dimensions.

6.2.1 *Twin peaks*

PolyChord is capable of clustering posteriors in very high dimensions. We define a twin peaks likelihood as an equal mixture of two spherical Gaussians, separated by a distance of 10σ.

PolyChord correctly identifies these clusters in arbitrary dimensions (tested up to $D = 100$), providing that n_{live} and n_{repeats} are scaled in proportion to D. It calculates a global evidence that agrees with the analytic results. In addition, the local evidences correctly divide the peaks in proportion to their evidence contribution.

The results for a twin peaks likelihood are of an identical character to Figures 6 & 7, and hence not included.

6.2.2 *Rastrigin function*

PolyChord's clustering capacity is very effective on complicated clustering problems as well. The n-dimensional Rastrigin test function is defined by:

$$f(\theta) = An + \sum_{i=1}^{n} \left[\theta_i^2 - A\cos(2\pi\theta_i)\right], \qquad (21)$$
$$A = 10, \qquad \theta_i \in [-5.12, 5.12].$$

This is the industry standard "bunch of grapes", the two-dimensional version of which is illustrated in Figure 8. For our purposes, we will treat (21) as the negative log-likelihood so that $\mathcal{L}(\theta) \propto \exp[-f(\theta)]$. This is a stereotypically hard problem to solve, as many algorithms get stuck in local maxima.

We ran PolyChord on a two-dimensional Rastrigin log-likelihood with $n_{\text{live}} = 1000$ and $n_{\text{repeats}} = 6$. With these settings, PolyChord calculates accurate evidence and posterior samples (Figure 8), and in addition correctly iso-

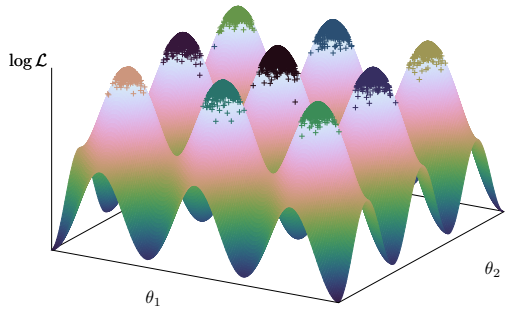

Figure 8. The two-dimensional Rastrigin log-likelihood in the range $[-1.5, 1.5]^2$. Within this region there are 8 local maxima, and one global maximum at $(0, 0)$. The clustered samples produced by PolyChord are plotted on the log-likelihood surface, with colours that indicating the separate clusters identified.

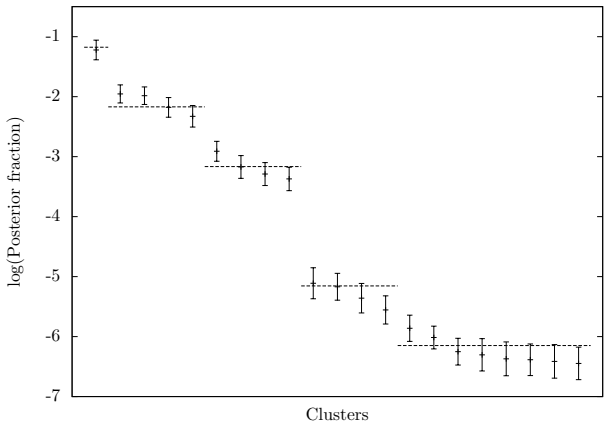

Figure 9. PolyChord cluster identification for the Rastrigin function. PolyChord identifies posterior modes and computes their local evidences, expressed here as a logarithmic fraction of the total evidence in the mode. Dashed lines indicate the analytic results computed by a saddle point approximation at each of the peaks. As can be seen, PolyChord reliably identifies the inner 21 modes with increasing accuracy.

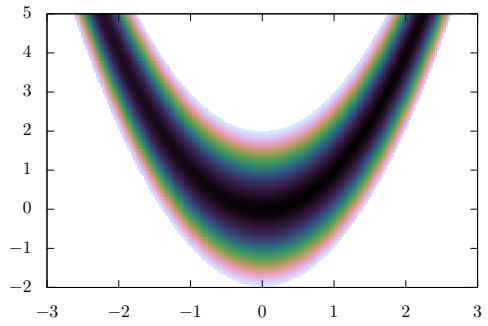

Figure 10. Density plot of the two-dimensional Rosenbrock function. The function exhibits a long, thin curving degeneracy, with a global maximum at $(1, 1)$.

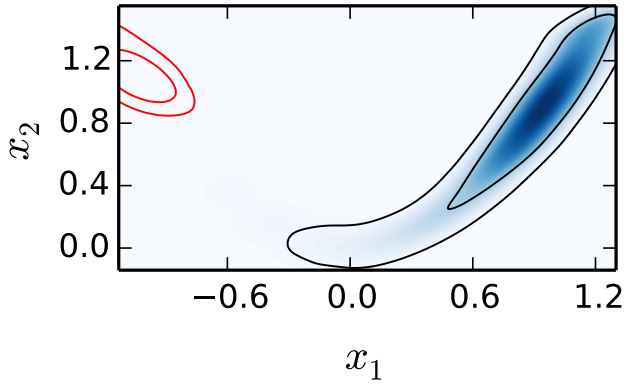

Figure 11. The four-dimensional Rosenbrock posterior, with x_3 and x_4 marginalised out. PolyChord correctly identifies both the local (red) and global (blue) maxima.

lates and computes local evidences for the inner 21 modes. Additional outer modes are also found, but these are combinations of lower modes due to their very low posterior fraction. Increasing the resolution parameter $n_{\rm live}$ further increases the number of modes identified. Examples of clustered posterior samples are indicated in Figure 9, coloured using Green's (2011) 'cubehelix'.

6.3 Rosenbrock function

PolyChord is also capable of navigating moderate curving degeneracies.

The n-dimensional Rosenbrock function is defined by:

$$f(x) = \sum_{i=1}^{n-1} (a - x_i)^2 + b(x_{i+1} - x_i^2)^2, \tag{22}$$

$$a = 1, \quad b = 100, \quad x_i \in [-5, 5], \tag{23}$$

the two-dimensional version of which is plotted in Figure 10. This is the industry standard "banana", as it exhibits an extremely long and flat curving degeneracy. We consider $n = 4$, in which there is a global maximum at $(1, 1, 1, 1)$ and a local maximum at $(-1, 1, 1, 1)$. The true evidence value is -15.1091, and with $n_{\rm live} = 1000, n_{\rm repeats} = 12$, PolyChord reliably finds both peaks (Figure 11) and produces a correct evidence estimation.

In higher dimensions, PolyChord reliably finds the local and global maxima. The lack of an analytic evidence value for the Rosenbrock function prevents a verification of the evidence calculation.

6.4 Gaussian shells

A "Gaussian shell" with mean μ, radius r and width w is defined as:

$$\log \mathcal{L}_{\rm shell}(\mathbf{x}|\mu, r, w) = A - \frac{(|\mathbf{x} - \mu| - r)^2}{2w^2}, \tag{24}$$

where A is a normalisation constant that may be calculated using a saddle point approximation. This likelihood is centered on some mean vector μ, and has a radial Gaussian profile with width w at distance r from this centre. This radial profile is then revolved around μ to create a spherical shell-like likelihood. A two-dimensional version of this likelihood is indicated in Figure 12.

This distribution may be representative of likelihoods that one may encounter in beyond-the-Standard-Model paradigms in particle physics. In such models, the majority of the posterior mass lies in thin sheets or hypersurfaces through the parameter space.

Running PolyChord on a 100-dimensional Gaussian shell with $n_{\rm live} = 1000$, $n_{\rm repeats} = 200$ yields consistent evidences and posteriors, shown in Figure 13.

Given that this problem is quoted as being "optimally difficult" (Feroz et al. 2009), the ease with which PolyChord tackles this problem in high dimensions is worth explanation. In the two-dimensional case, it is clear that the posterior mass is concentrated in a very thin, curving region of the parameter space. However, as the dimensionality is increased, more and more of the n-sphere's volume is concentrated at the edge, and the thin characteristic of the degeneracy is lost.

This may mean that the Gaussian shell is not a good proxy for a high-dimensional curving degeneracy. However, it could equally suggest that curving degeneracies become easier to navigate in higher dimensions. We can certainly conclude that a particle physics model with a proliferation of phases would be easier to navigate than one with a smaller number of phases.

6.4.1 Twin Gaussian shells

We finish our toy problems by combining the difficulties of multimodality (Section 6.2) and degeneracy, by mixing two twin Gaussian shells together:

$$\mathcal{L}(\mathbf{x}) \propto \mathcal{L}_{\rm shell}(\mathbf{x}|\mu_1, r, w) + \mathcal{L}_{\rm shell}(\mathbf{x}|\mu_2, r, w). \tag{25}$$

We choose $r = 2$, $w = 0.1$, and μ_1 and μ_2 are separated by 7 units. With $n_{\rm live} = 10 n_{\rm dims}$ and $n_{\rm repeats} = 2 n_{\rm dims}$, PolyChord successfully computes the local and global posteriors and evidences up to $D = 100$, and reliably identifies the two modes. The comparison of run times with MultiNest recovers a similar pattern to Figure 7, although in our experience, the MultiNest parameters require some tuning to ensure that evidences are calculated correctly when $n_{\rm dims} > 30$.

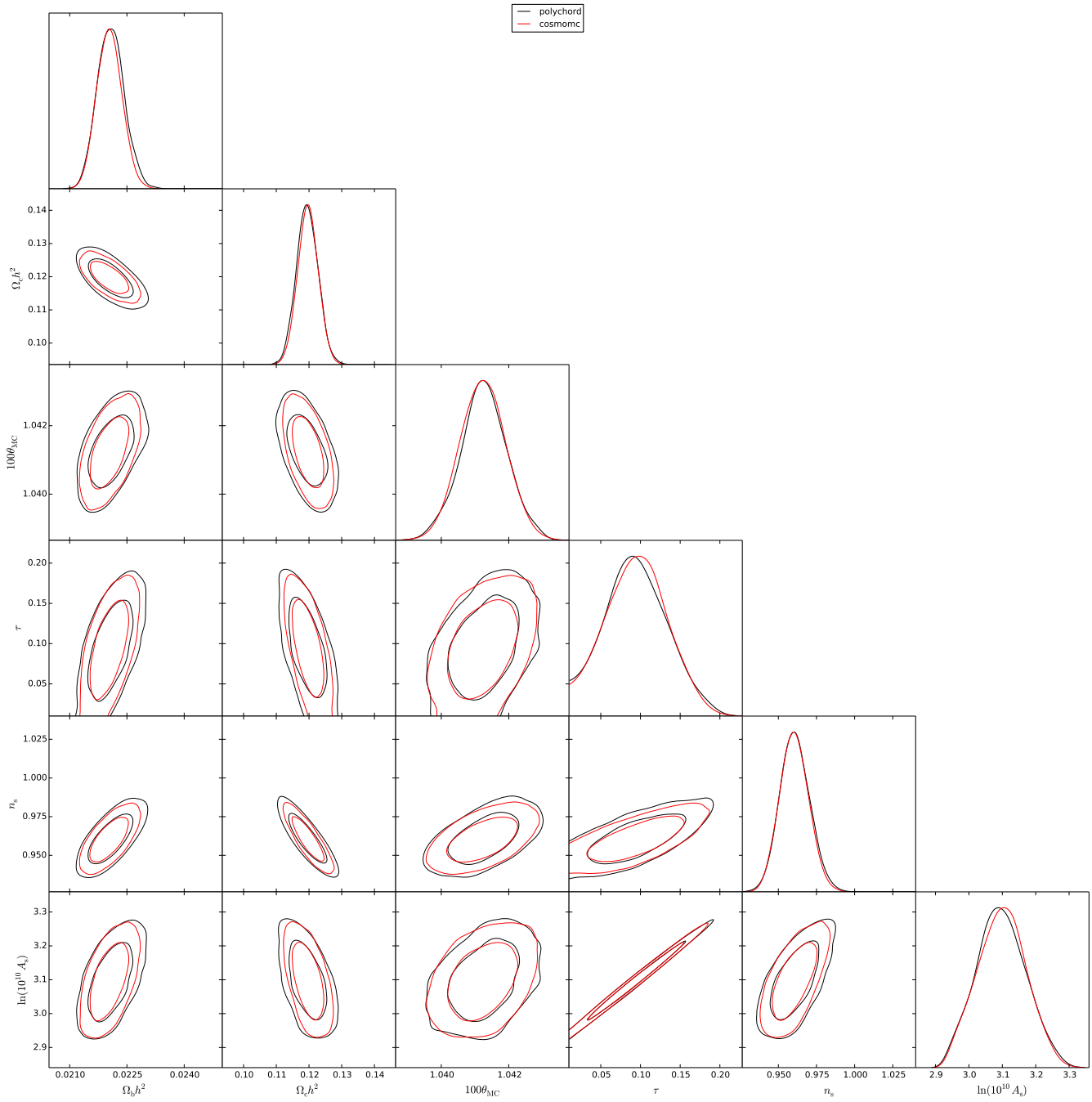

Figure 14. CosmoChord (red) vs. CosmoMC (black). We use the 2013 `CAMSPEC+commander` likelihoods with a standard six-parameter ΛCDM cosmology, varying all 14 nuisance parameters. We compare the 1 and 2-dimensional marginalised posteriors of the 6 ΛCDM parameters. CosmoChord is in close agreement with the posteriors produced by CosmoMC, recovering the correct mean values of and degeneracies between the parameters. The slight deviations between the red and black curves are sampling noise.

6.5 CosmoChord

An additional strength of PolyChord lies in its ability to exploit a fast-slow hierarchy common in many cosmological applications.

As an example, we consider the likelihoods provided by CAMB (Lewis et al. 2000) and CosmoMC (Lewis & Bridle 2002). In Boltzmann codes such as CAMB, parameters controlling the primordial power spectrum (such as n_s and A_s) do not require recalculation of transfer functions. These parameters are termed "semi-slow". In addition, modern Planck likelihoods (Planck Collaboration et al. 2014) have nuisance parameters associated with the foregrounds. These may be varied without recalculation of the cosmological background. These parameters are hence termed "fast". CosmoMC (Lewis & Bridle 2002) implements this hierarchy of speeds in its likelihood calculation.

We have successfully implemented PolyChord within CosmoMC, and term the result CosmoChord. The tra-

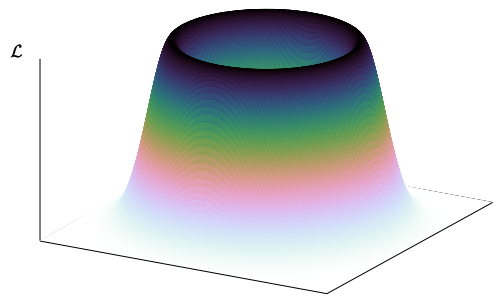

Figure 12. The two-dimensional Gaussian shell likelihood.

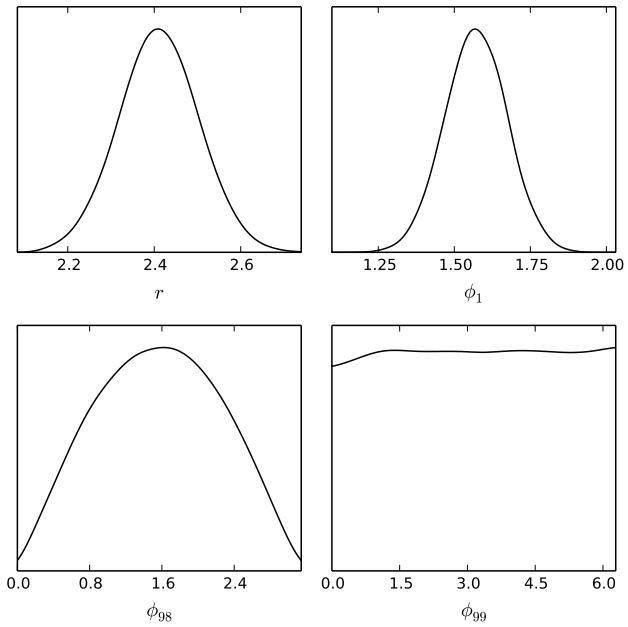

Figure 13. Posteriors produced by PolyChord for a $n = 100$-dimensional Gaussian shell, with width $w = 0.1$, radius $r = 2$, and center $\mu = \mathbf{0}$. Plotting the marginalised posteriors for the Cartesian sampling parameters $\{x_1, \cdots, x_n\}$ yields Gaussian distributions centered on the origin. To see the effectiveness of the sampler it is better to plot the sampling parameters in terms of n-dimensional spherical polar coordinates $\{r, \phi_1, \cdots, \phi_{n-1}\}$. Note that the polar coordinates are *derived parameters*, and that the sampling space still has the strong Gaussian shell degeneracy. In this case we can see that the radial coordinate has a Gaussian profile centered on $r_0 = r \times \frac{1}{2}\left(1 + \sqrt{1 + 4(n-1)(w/r)^2}\right)$ with width $w_0 = w(1 + (n-1)(w/r_0)^2)^{-1/2}$. The azimuthal coordinate ϕ_{n-1} has a uniform posterior, and the other angular coordinates $\{\phi_i\}$ have posteriors defined by $\mathrm{P}(\phi_i) \propto (\sin\phi_i)^{n-i-1}$.

ditional Metropolis–Hastings algorithm is replaced with nested sampling. This implementation is available to download from the link at the end of the paper.

The exploitation of fast-slow parameters means that CosmoChord vastly outperforms MultiNest when running with modern Planck likelihoods.

CosmoMC by default uses a Metropolis–Hastings sampler. If this has a well-tuned proposal distribution (e.g. if one is performing importance sampling from an already well-characterised likelihood), then PolyChord is 2–4 times slower than the traditional CosmoMC. If proposal matrices are unavailable (e.g. in the case that one is examining an entirely new model) then CosmoChord's run time is competitive with the native CosmoMC sampler. This is a good example of the self-tuning capacity of PolyChord, since it only requires two tuning parameters, as opposed to $\sim \mathcal{O}(D^2)$.

CosmoChord produces parameter estimations consistent with CosmoMC (Figure 14). It has been implemented effectively in multiple cosmological applications in the latest Planck paper describing constraints on inflation (Planck Collaboration XX 2015), including application to a 37-parameter reconstruction problem (4 slow, 19 semi-slow, 14 fast). In addition, PolyChord is an integral component of the ModeChord code, a combination of CosmoChord and ModeCode (Mortonson et al. 2011; Easther & Peiris 2012; Norena et al. 2012), which is available at http://modecode.org/.

7 CONCLUSIONS

We have introduced PolyChord, a novel nested sampling algorithm tailored for high-dimensional parameter spaces. It is able to fully exploit a hierarchy of parameter speeds such as is found in CosmoMC and CAMB. It utilises slice sampling at each iteration to sample within the hard likelihood constraint of nested sampling. It can identify and evolve separate modes of a posterior semi-independently and is parallelised using openMPI.

ACKNOWLEDGEMENTS

We would like to thank Farhan Feroz for numerous helpful discussions during the inception of the PolyChord algorithm. W H thanks STFC for their support.

DOWNLOAD LINK

PolyChord is available for download from:
http://ccpforge.cse.rl.ac.uk/gf/project/polychord/

APPENDIX A: PRIOR TRANSFORMATIONS

Here we give examples of the procedure for calculating the transformation from the unit hypercube to the physical space. We demonstrate it for a simple separable case, and a more complicated dependent case

To recap, we aim to compute the inverse of the functions F_i:

$$F_i(\theta_i|\theta_{i-1},\ldots,\theta_0) = \int_0^{\theta_i} \pi_i(\theta_i'|\theta_{i-1},\ldots,\theta_1)d\theta_i', \tag{A1}$$

where:

$$\pi_i(\theta_i|\theta_{i-1},\ldots,\theta_0) = \frac{\int \pi_i(\theta)d\theta_{i+1}\ldots d\theta_N}{\int \pi_i(\theta)d\theta_i\ldots d\theta_N}. \tag{A2}$$

$\mathbf{F}$ maps from θ in the physical space onto the unit hypercube injectively.

A1 Separable priors

A separable prior satisfies:

$$\pi(\theta) = \prod_i \pi_i(\theta_i). \tag{A3}$$

This has the fortunate side effect that the functions F_i only depend on θ_i:

$$F_i(\theta_i|\theta_{i-1},\ldots,\theta_0) = F_i(\theta_i). \tag{A4}$$

Solving a separable prior thus amounts to solving a one-dimensional inverse-transform sampling problem. We demonstrate this procedure for two cases, a rectangular uniform prior, and a Gaussian prior.

A1.1 Uniform prior

A rectangular uniform prior is defined by two parameters, $\theta_{\min}, \theta_{\max}$:

$$\pi(\theta) = \begin{cases} (\theta_{\max}-\theta_{\min})^{-1} & \text{for } \theta_{\max} < \theta_i < \theta_{\min} \\ 0 & \text{otherwise.} \end{cases} \tag{A5}$$

Computing $F(\theta)$ we find:

$$\begin{aligned} F(\theta) &= \int_{-\infty}^{\theta} \pi(\theta')d\theta', \\ &= \frac{\theta-\theta_{\min}}{\theta_{\max}-\theta_{\min}}, \end{aligned} \tag{A6}$$

with $F = 0$ or 1 either side of $\theta_{\min}$ and $\theta_{\max}$ respectively. Inverting the equation $F(\theta) = x$ we find:

$$\theta = \theta_{\min} + (\theta_{\max}-\theta_{\min})x, \tag{A7}$$

is the transformation from x in the unit hypercube to θ in the physical space.

A1.2 Gaussian prior

Defining a Gaussian prior with mean μ and standard deviation σ:

$$\pi(\theta) = \frac{1}{\sqrt{2\pi}\sigma}\exp\left[-\frac{(x-\mu)^2}{2\sigma^2}\right], \tag{A8}$$

We find that the procedure above yields:

$$\theta = \mu + \sqrt{2}\sigma\text{erfinv}(2x-1), \tag{A9}$$

where erfinv is the conventional inverse error function.

A2 Forced identifiability priors

As an example of a prior that is not separable in the parameters, we consider a forced identifiability prior. Here, n parameters are distributed uniformly between $\theta_{\min}$ and $\theta_{\max}$, but subject to the constraint that they are ordered numerically. This is a particularly useful prior in the reconstruction of functions using a spline with movable knots (Vázquez et al. 2012; Aslanyan et al. 2014; Abazajian et al. 2014; Planck Collaboration XX 2015). In this case, the horizontal locations of the knots must be ordered.

The required prior is uniform in the hyper-triangle defined by $\theta_{\min} < \theta_1 < \cdots < \theta_n < \theta_{\max}$, and zero everywhere else:

$$\pi(\theta) = \begin{cases} \frac{1}{n!(\theta_{\max}-\theta_{\min})^n} & \text{for } \theta_{\min} < \theta_1 < \cdots < \theta_n < \theta_{\max} \\ 0 & \text{otherwise.} \end{cases} \tag{A10}$$

To calculate equations (A1 & A2) we simply integrate over the constant distribution, taking care with the limits. We find:

$$\pi_i(\theta_i|\theta_{i-1},\ldots,\theta_0) = \frac{(n-i+1)(\theta_i-\theta_{i-1})^{n-i}}{(\theta_{\max}-\theta_{\min})^{n-i+1}}, \tag{A11}$$

$$F_i(\theta_i|\theta_{i-1},\ldots,\theta_0) = \left(\frac{\theta_i-\theta_{i-1}}{\theta_{\max}-\theta_{i-1}}\right)^{n-i+1}, \tag{A12}$$

where for consistency we define $\theta_0 = \theta_{\min}$. Hence solving $x_i = F(\theta_i|\theta_{i-1},\ldots,\theta_0)$ for θ_i we find:

$$\theta_i = \theta_{i-1} + (\theta_{\max}-\theta_{i-1})x_i^{1/(n-i+1)}. \tag{A13}$$

This enables $\{\theta_i\}$ to be calculated sequentially from $\{x_i\}$. We may interpret this transformation as θ_i being distributed as the smallest of $n-i+1$ uniformly distributed variables in the range $[\theta_{i-1}, \theta_{\max}]$.

APPENDIX B: EVIDENCE ESTIMATES AND ERRORS

Skilling (2006) initially advocated using Monte-Carlo methods to estimate the evidence error, although this requires the storage of the entire chain of dead points, rather than just the subset usually stored for posterior inferences. For high-dimensional problems, the number of dead points is prohibitively large, and cannot be stored.

Feroz et al. (2009) use an alternative method based on the relative entropy (also suggested by Skilling (2006)).

Keeton (2011) suggests a more intuitive methodology of estimating the error, and it is this which we use, although it must be heavily adapted for the case of variable numbers of live points and clustering.

B1 Basic theory

We wish to compute the sum:

$$\mathcal{Z} = \sum_i (X_{i-1}-X_i)\mathcal{L}_i. \tag{B1}$$

However, we do not know the volumes X_i exactly, so we can only make inferences about $\mathcal{Z}$, in terms of a probability

distribution $\mathrm{P}(\mathcal{Z})$. In practice, all we need to compute is the mean and variance of this distribution:

$$\mathrm{mean}(\mathcal{Z}) \equiv \overline{\mathcal{Z}}, \tag{B2}$$

$$\mathrm{var}(\mathcal{Z}) \equiv \overline{\mathcal{Z}^2} - \overline{\mathcal{Z}}^2. \tag{B3}$$

At iteration i, the $n_{\rm live}$ live points are each uniformly sampled within a contour of volume X_{i-1}. The volume X_i will be the largest volume out of $n_{\rm live}$ uniform volume samples in volume X_i. Thus X_i satisfies the recursion relation:

$$X_i = tX_{i-1}, \qquad X_0 = 1, \tag{B4}$$

$$P(t) = n_{\rm live} t^{n_{\rm live}-1}, \tag{B5}$$

where the t and X_{i-1} are independent.

It is worth noting that the procedure described below will generate the mean and variance of the distribution, but in fact this is not quite what we want. The evidence is in practice approximately log-normally distributed. Thus, it is better to report the mean and variance of $\log \mathcal{Z}$, defined by:

$$\mathrm{mean}(\log \mathcal{Z}) = 2\log\overline{\mathcal{Z}} - \frac{1}{2}\log\overline{\mathcal{Z}^2}, \tag{B6}$$

$$\mathrm{var}(\log \mathcal{Z}) = \log\overline{\mathcal{Z}^2} - 2\log\overline{\mathcal{Z}}. \tag{B7}$$

B2 Computing the mean evidence

While it is possible to take equations (B1,B4 & B5) and compute the mean as a general formula (Keeton 2011), in the case of clustering this is uninformative. In fact, for large-dimensional spaces using the full formula would require storage of a prohibitively large amount of data. The calculation is better accomplished by a set of recursion relations, which update the mean evidence and its error at each step.

For now, assume that we have n live points currently enclosed by some likelihood contour $\mathcal{L}$ of volume X, and $\mathcal{Z}$ is the last value of the evidence calculated from all of the points that have died so far. By considering (B1,B4&B5), when we kill off the outermost point, we may adjust the values of $\mathcal{Z}$ and X using:

$$\mathcal{Z} \to \mathcal{Z} + (1-t)X\mathcal{L}, \tag{B8}$$

$$X \to tX. \tag{B9}$$

Taking the mean of these relations, we may use the facts that t and X are independent random variables and that $P(t) = nt^{n-1}$, to find the recursion relations:

$$\overline{\mathcal{Z}} \to \overline{\mathcal{Z}} + \frac{1}{n+1}\overline{X}\mathcal{L}, \tag{B10}$$

$$\overline{X} \to \frac{n}{n+1}\overline{X}. \tag{B11}$$

B3 Computing the evidence error

To estimate $\overline{\mathcal{Z}^2}$, we square (B8) and (B9) and multiply both together to obtain:

$$\mathcal{Z}^2 \to \mathcal{Z}^2 + 2(1-t)\mathcal{Z}X\mathcal{L} + (1-t)^2X^2\mathcal{L}^2, \tag{B12}$$

$$\mathcal{Z}X \to t\mathcal{Z}X + t(1-t)X^2\mathcal{L}, \tag{B13}$$

$$X^2 \to t^2X^2. \tag{B14}$$

Note that we now need to keep track of the variable $\mathcal{Z}X$, as these two are not independent. Taking the averages of the above yields:

$$\overline{\mathcal{Z}^2} \to \overline{\mathcal{Z}^2} + \frac{2\overline{\mathcal{Z}X}\mathcal{L}}{n+1} + \frac{2\overline{X^2}\mathcal{L}^2}{(n+1)(n+2)}, \tag{B15}$$

$$\overline{\mathcal{Z}X} \to \frac{n\overline{\mathcal{Z}X}}{n+1} + \frac{n\overline{X^2}\mathcal{L}}{(n+1)(n+2)}, \tag{B16}$$

$$\overline{X^2} \to \frac{n}{n+2}\overline{X^2}. \tag{B17}$$

B4 The full calculation

There are therefore five quantities to keep track of:

$$\overline{\mathcal{Z}}, \quad \overline{\mathcal{Z}^2}, \quad \overline{\mathcal{Z}X}, \quad \overline{X}, \quad \overline{X^2}.$$

These should be initialised at $\{0,0,0,1,1\}$ respectively, and updated using equations (B10,B12,B13,B11,B14) in that order. In fact, we keep track of the logarithm of these quantities, in order to avoid machine precision errors.

APPENDIX C: EVIDENCE ESTIMATES AND ERRORS IN CLUSTERS

This analysis follows that of Appendix B. We recommend that you have understood the methods described there before continuing.

Throughout the algorithm, there will in general be m identified clusters. In doing so, we wish to keep track of the volume of each cluster $\{X_1, \ldots, X_m\}$, the global evidence and its error $\mathcal{Z}, \mathcal{Z}^2$ and the local evidences and their errors $\{\mathcal{Z}_1, \mathcal{Z}_1^2 \ldots, \mathcal{Z}_m, \mathcal{Z}_m^2\}$. At each iteration, the point with the lowest likelihood $\mathcal{L}$ will be killed from cluster p, $(1 \leqslant p \leqslant m)$.

C1 Evidence

We thus need to update the global evidence, the local evidence of cluster p, and the volume of cluster p:

$$\mathcal{Z} \to \mathcal{Z} + (1-t)X_p\mathcal{L}, \tag{C1}$$

$$\mathcal{Z}_p \to \mathcal{Z}_p + (1-t)X_p\mathcal{L}, \tag{C2}$$

$$X_p \to tX_p. \tag{C3}$$

Since t will be distributed with $P(t) = n_p t^{n_p-1}$, taking the mean of these yields:

$$\overline{\mathcal{Z}} \to \overline{\mathcal{Z}} + \frac{\overline{X}_p\mathcal{L}}{n_p+1}, \tag{C4}$$

$$\overline{\mathcal{Z}}_p \to \overline{\mathcal{Z}}_p + \frac{\overline{X}_p\mathcal{L}}{n_p+1}, \tag{C5}$$

$$\overline{X}_p \to \frac{n_p\overline{X}_p}{n_p+1}. \tag{C6}$$

Keeping track of $\{\overline{\mathcal{Z}}, \overline{\mathcal{Z}_p}, \overline{X_p}, p = 1 \ldots m\}$ and updating them using the recursion relations in the order above will produce a consistent evidence estimate for both the local and global evidence errors.

C2 Evidence errors

We must also keep track of the local and global evidence errors. Taking the square of equations (C1 & C2) yields:

$$\mathcal{Z}^2 \to \mathcal{Z}^2 + 2(1-t)\mathcal{Z}X_p\mathcal{L} + (1-t)^2X_p^2\mathcal{L}^2, \tag{C7}$$

$$\mathcal{Z}_p^2 \to \mathcal{Z}_p^2 + 2(1-t)\mathcal{Z}_pX_p\mathcal{L} + (1-t)^2X_p^2\mathcal{L}^2. \tag{C8}$$

We can see that we're going to need to keep track of $\{\overline{\mathcal{Z}X_p}, \overline{\mathcal{Z}_pX_p}, \overline{X_p^2}\}$ in addition to $\{\overline{\mathcal{Z}^2}, \overline{\mathcal{Z}_p^2}\}$. Taking various multiplications of equations (C1, C2 & C3) finds:

$$\mathcal{Z}X_p \to t\mathcal{Z}X_p + (1-t)tX_p^2\mathcal{L}, \tag{C9}$$

$$\mathcal{Z}X_q \to \mathcal{Z}X_q + (1-t)X_pX_q\mathcal{L} \qquad (p \neq q), \tag{C10}$$

$$\mathcal{Z}_pX_p \to t\mathcal{Z}_pX_p + (1-t)tX_p^2\mathcal{L}, \tag{C11}$$

$$X_p^2 \to t^2X_p^2, \tag{C12}$$

$$X_pX_q \to tX_pX_q. \tag{C13}$$

Taking the mean of the above yields the recursion relations:

$$\overline{\mathcal{Z}^2} \to \overline{\mathcal{Z}^2} + \frac{2\overline{\mathcal{Z}X_p}\mathcal{L}_p}{n_p+1} + \frac{2\overline{X_p^2}\mathcal{L}^2}{(n_p+1)(n_p+2)}, \tag{C14}$$

$$\overline{\mathcal{Z}_p^2} \to \overline{\mathcal{Z}_p^2} + \frac{2\overline{\mathcal{Z}_pX_p}\mathcal{L}}{n_p+1} + \frac{2\overline{X_p^2}\mathcal{L}^2}{(n_p+1)(n_p+2)}, \tag{C15}$$

$$\overline{\mathcal{Z}X_p} \to \frac{n_p\overline{\mathcal{Z}X_p}}{n_p+1} + \frac{n_p\overline{X_p^2}\mathcal{L}}{(n_p+1)(n_p+2)}, \tag{C16}$$

$$\overline{\mathcal{Z}X_q} \to \overline{\mathcal{Z}X_p} + \frac{\overline{X_pX_q}\mathcal{L}}{(n_p+1)} \qquad (q \neq p), \tag{C17}$$

$$\overline{\mathcal{Z}_pX_p} \to \frac{n_p\overline{\mathcal{Z}_pX_p}}{n_p+1} + \frac{n_p\overline{X_p^2}\mathcal{L}}{(n_p+1)(n_p+2)}, \tag{C18}$$

$$\overline{X_p^2} \to \frac{n_p\overline{X_p^2}}{n_p+2}, \tag{C19}$$

$$\overline{X_pX_q} \to \frac{n_p\overline{X_pX_q}}{n_p+1}. \tag{C20}$$

Keeping track of

$$\{\overline{\mathcal{Z}^2}, \overline{\mathcal{Z}_p^2}, \overline{\mathcal{Z}X_p}, \overline{\mathcal{Z}_pX_p}, \overline{X_p^2}, \overline{X_pX_q}, p, q = 1\ldots m\},$$

and updating them using the recursion relations in the order above will produce a consistent estimate for the local and global evidence errors.

C3 Cluster initialisation

All that remains is to initialise the clusters correctly at the point of creation.

The starting initialisation of the evidence and volume is reasonable, there will be only a single cluster with volume 1, and all evidence related terms 0. At some point (possibly at the beginning, depending on the prior), the live points will split into distinct clusters, and the local volumes and evidences will need to be re-initialised.

At the point of splitting a cluster into sub-clusters, we partition the n live points into a N new clusters, with $\{n_1, \ldots, n_N\}$ live points in each. If the volume of the splitting cluster is X_p initially, we need to know how to partition this volume into $\{X_1, \ldots, X_N\}$. If the points are drawn uniformly from the volume, then the n_i will depend on the volumes via a multinomial probability distribution:

$$P(\{n_i\}|X_p, \{X_i\}) \propto {X_1}^{n_1} \ldots X_N^{n_N}. \tag{C21}$$

We however want to know the probability distributions of the $\{X_i\}$, given the $\{n_i\}$. We can invert the above with Bayes' theorem, using an (improper) logarithmic prior on the volumes subject to the constraint that they sum to X_p:

$$P(\{X_i\}|X_p) \propto \frac{\delta(X_1 + \cdots + X_N - X_p)}{X_1 \cdots X_N}. \tag{C22}$$

Doing this shows the posterior $P(\{X_i\}|X_p, \{n_i\})$ is a Dirichlet distribution with parameters $\{n_i\}$. More importantly, we can use this to compute the means and correlations for the volumes $\{X_i\}$:

$$\overline{X_i} = \frac{n_i}{n}\overline{X}_p, \tag{C23}$$

$$\overline{X_i^2} = \frac{n_i(n_i+1)}{n(n+1)}\overline{X_p^2}, \tag{C24}$$

$$\overline{X_iX_j} = \frac{n_in_j}{n(n+1)}\overline{X_p^2}, \tag{C25}$$

$$\overline{X_iY} = \frac{n_i}{n}\overline{X_pY} \qquad Y \in \{Z, Z_p, X_q\}. \tag{C26}$$

The first equation recovers the intuitive result that the volume should split as the fraction of live points. Note, however that this requires a logarithmic prior. The third shows us that since $\overline{X_iX_j} \neq \overline{X_i}\,\overline{X_j}$, the volumes are correlated at the splitting. This is to be expected.

We also need to initialise the local evidences and their errors. A consistent approach is to assume that the evidences also split in proportion to the cluster distribution of live points. Following the same reasoning as above, we find that:

$$\overline{Z_i} = \frac{n_i}{n}\overline{Z}_p \tag{C27}$$

$$\overline{Z_iX_i} = \frac{n_i(n_i+1)}{n(n+1)}\overline{Z_pX_p} \tag{C28}$$

$$\overline{Z_i^2} = \frac{n_i(n_i+1)}{n(n+1)}\overline{Z_p^2} \tag{C29}$$

$$\tag{C30}$$

Thus, at cluster splitting, all of the new local evidences, volumes and cross correlations are initialised according to the above.

This completes the mechanism for keeping track of the local and global evidences, their errors, and the local cluster volumes.

REFERENCES

Abazajian, K. N., Aslanyan, G., Easther, R., & Price, L. C. 2014, ArXiv e-prints, arXiv:1403.5922

Aitken, S., & Akman, O. 2013, BMC Systems Biology, 7, doi:10.1186/1752-0509-7-72

Aslanyan, G., Price, L. C., Abazajian, K. N., & Easther, R. 2014, ArXiv e-prints, arXiv:1403.5849

Betancourt, M. 2011, in American Institute of Physics Conference Series, Vol. 1305, American Institute of Physics Conference Series, ed. A. Mohammad-Djafari, J.-F. Bercher, & P. Bessiére, 165–172

Brewer, B. J., Pártay, L. B., & Csányi, G. 2009, ArXiv e-prints, arXiv:0912.2380

Easther, R., & Peiris, H. V. 2012, Phys.Rev., D85, 103533

Feroz, F., & Hobson, M. P. 2008, MNRAS, 384, 449

Feroz, F., Hobson, M. P., & Bridges, M. 2009, MNRAS, 398, 1601

Feroz, F., Hobson, M. P., Cameron, E., & Pettitt, A. N. 2013, ArXiv e-prints, arXiv:1306.2144

Feroz, F., & Skilling, J. 2013, in American Institute of Physics Conference Series, Vol. 1553, American Institute of Physics Conference Series, ed. U. von Toussaint, 106–113

Green, D. A. 2011, Bulletin of the Astronomical Society of India, 39, 289
Handley, W. J., Hobson, M. P., & Lasenby, A. N. 2015, ArXiv e-prints, arXiv:1502.01856
Keeton, C. R. 2011, MNRAS, 414, 1418
Lewis, A. 2013, Phys. Rev. D, 87, 103529
Lewis, A., & Bridle, S. 2002, Phys. Rev. D, 66, 103511
Lewis, A., Challinor, A., & Lasenby, A. 2000, Astrophys. J., 538, 473
MacKay, D. J. C. 2002, Information Theory, Inference & Learning Algorithms (New York, NY, USA: Cambridge University Press)
Mortonson, M. J., Peiris, H. V., & Easther, R. 2011, Phys.Rev., D83, 043505
Mukherjee, P., Parkinson, D., & Liddle, A. R. 2006, ApJ, 638, L51
Neal, R. M. 2000, ArXiv Physics e-prints, physics/0009028
Norena, J., Wagner, C., Verde, L., Peiris, H. V., & Easther, R. 2012, Phys.Rev., D86, 023505
Planck Collaboration, Ade, P. A. R., Aghanim, N., et al. 2014, A&A, 571, A15
Planck Collaboration XX. 2015, ArXiv e-prints, arXiv:1502.02114
Sivia, D. S., & Skilling, J. 2006, Data analysis : a Bayesian tutorial, Oxford science publications (Oxford, New York: Oxford University Press)
Skilling, J. 2006, Bayesian Analysis, 1, 833
Vázquez, J. A., Bridges, M., Hobson, M. P., & Lasenby, A. N. 2012, J. Cosmology Astropart. Phys., 6, 6