GRID-BASED EXPLORATION OF COSMOLOGICAL PARAMETER SPACE WITH SNAKE

K. MIKKELSEN\textsuperscript{1}, S. K. NÆSS\textsuperscript{1}, AND H. K. ERIKSEN\textsuperscript{1,2}

\textsuperscript{1} Institute of Theoretical Astrophysics, University of Oslo, P.O. Box 1029, Blindern, NO-0315 Oslo, Norway; kristin.mikkelsen@astro.uio.no
\textsuperscript{2} Centre of Mathematics for Applications, University of Oslo, P.O. Box 1053, Blindern, NO-0316 Oslo, Norway

Received 2012 November 13; accepted 2013 September 23; published 2013 October 24

ABSTRACT

We present a fully parallelized grid-based parameter estimation algorithm for investigating multidimensional likelihoods called Snake, and apply it to cosmological parameter estimation. The basic idea is to map out the likelihood grid-cell by grid-cell according to decreasing likelihood, and stop when a certain threshold has been reached. This approach improves vastly on the “curse of dimensionality” problem plaguing standard grid-based parameter estimation simply by disregarding grid cells with negligible likelihood. The main advantages of this method compared to standard Metropolis–Hastings Markov Chain Monte Carlo methods include (1) trivial extraction of arbitrary conditional distributions; (2) direct access to Bayesian evidences; (3) better sampling of the tails of the distribution; and (4) nearly perfect parallelization scaling. The main disadvantage is, as in the case of brute-force grid-based evaluation, a dependency on the number of parameters, \( N_{\text{par}} \). One of the main goals of the present paper is to determine how large \( N_{\text{par}} \) can be, while still maintaining reasonable computational efficiency; we find that \( N_{\text{par}} = 12 \) is well within the capabilities of the method. The performance of the code is tested by comparing cosmological parameters estimated using Snake and the WMAP-7 data with those obtained using CosmoMC, the current standard code in the field. We find fully consistent results, with similar computational expenses, but shorter wall time due to the perfect parallelization scheme.

Key words: cosmic background radiation – cosmology: observations – methods: statistical

Online-only material: color figures

1. INTRODUCTION

Cosmological models are described in terms of a modest number of cosmological parameters that reflect the underlying physical processes of the universe. These are today routinely measured by experiments such as the Wilkinson Microwave Anisotropy Probe (WMAP; Jarosik et al. 2011), Planck Collaboration (2011) and the Sloan Digital Sky Survey (York et al. 2000) through likelihood techniques.

The most popular parameter estimation algorithm in the cosmology community to date is the CosmoMC package (Lewis & Bridle 2002), which maps out the cosmological parameter space using a Metropolis–Hastings Markov Chain Monte Carlo (MCMC) sampler. The computational cost of this method is almost exclusively determined by the external book-keeping operation: the expense of the internal book-keeping operation is completely negligible compared to this. A complete analysis of current data sets typically requires \( O(10^5) \) evaluations, resulting in an overall computational cost of 100–10,000 CPU hours, depending on the particular problem.

This process can be sped up in two fundamentally different ways, namely either by reducing the cost per likelihood evaluation, or by reducing the number of likelihood evaluations required, and both cases have already been explored extensively in the literature. Examples of the former include CMBFit (Sandvik et al. 2004), PICO (Fendt & Wandelt 2007), COSMONET (Auld et al. 2007), sparse grids (Frommert et al. 2010), and PKANN (Agarwal et al. 2012), all of which essentially build up a library of known cosmological models given a set of parameters, and interpolate within this library using some statistical method. Examples of the latter include MultiNest (Feroz et al. 2009), APS (Daniel et al. 2012), CosmoPMC (Kilbinger et al. 2011), and HMC (Hajian 2007), all of which reduce the number of likelihood evaluations through more efficient sampling algorithms than the Metropolis–Hastings sampler.

In this paper, we present an algorithm that falls in the last category, aiming to reduce the total number of likelihood evaluations rather than the cost per evaluation. The initial idea of this paper is based on the following reasoning: if the problem under consideration involved only a one-dimensional likelihood, the mapping algorithm of choice would be obvious—one would simply evaluate the likelihood over a one-dimensional grid. The resulting function is both easier to work with than a set of samples, as produced by an MCMC algorithm, and more accurate. Furthermore, it generally requires fewer evaluations, because whereas an MCMC approach builds up the shape of the distribution by counting how many samples fall in a given parameter range (“bin”), the direct approach only needs to evaluate the likelihood in a given bin once. In other words, the MCMC approach spends most of the time evaluating the same likelihood points over and over again, which can give the direct evaluation approach a computational edge.

The vast majority of two-dimensional likelihoods are also mapped by grid methods rather than MCMC methods, while for three or four dimensions, the preferred approach is not clear. However, for higher dimensions, virtually all cases are so far handled by MCMC methods. At this stage, the so-called curse of dimensionality becomes highly relevant, as the number of likelihood evaluations depends exponentially on the number of dimensions. For instance, computing 100 grid points in each of five dimensions requires 100\(^5\) evaluation, which is generally far too many for most problems.

However, in this paper we point out that this is not necessarily true. The point is simply that the vast majority of the high-dimensionality volume typically has negligible likelihood, and therefore does not need to be evaluated in the first place. The trick is to figure out which grid cells are relevant and which
are not. If this can be done both efficiently and robustly, all the useful properties of normal grids are retained, and computational cost is not compromised. Further, by virtue of not being a Markov chain, the algorithm parallelizes trivially, leading to shorter overall computational wall time, which is often even more critical for a given analysis problem than the total CPU time.

2. THE SNAKE ALGORITHM

2.1. Algorithm

The Snake algorithm is very simple, and can easily be explained in terms of a few basic steps. To do so succinctly, it is useful to first define some terminology.

The grid. The Snake algorithm operates on a virtual grid in parameter space with the origin, \( \theta_0 \), set to the starting point selected by the user. This point together with the cell size in each dimension, \( \Delta \theta \), defines the position of every other grid point in parameter space through \( \theta_0 + k \cdot \Delta \theta \), where \( k \) is an integer vector describing the cell location with respect to the origin.

The surface. Each point on the grid is assigned to one of three groups depending on whether they are external, internal, or surface points. External points are those that have not yet been considered; internal points are those for which the point itself and all its neighbors have been visited; and surface points are the points that have been considered, but have at least one unexplored neighbor.

The repository. Considered parameter points are assigned a book-keeping integer, \( i \), representing that this point was the \( i \)th point visited by Snake. These points are stored as objects in a data structure called a repository, which is a two-dimensional dynamic list in which each row defines a point on the grid. In addition to the integer \( i \), each data row contains the point’s \( k \) vector, likelihood value \( L_i \), the integers corresponding to its neighbors, and a logical flag specifying whether the point is currently on the surface.

Given these definitions, the Snake algorithm may be summarized as follows.

1. Initialization. Compute the likelihood of the starting point, and insert the required information into the repository.
2. Neighbor. Consider the surface point with the highest likelihood value, which has index \( i_5 \), and randomly pick one of its unexplored neighbors. Evaluate the likelihood of this new point.
3. Surface update. Find the integers corresponding to all the neighbors of the new point. If any of these neighbors or the new point no longer have any unexplored neighbors, set its surface flag to false and remove it from the list of surface points.
4. Update repository. Insert the new point into the repository structure and update the relevant information for its neighbors.
5. Convergence check. The grid point with the overall highest likelihood, with \( i = i_{\text{peak}} \), and the current surface points are used to test if Snake has converged. If \( \log L(i_{\text{peak}}) - \log L(\theta(i)) \) is smaller than a predefined threshold for all surface points Snake has converged and the routine exits. Otherwise Snake loops back to (2) and performs another evaluation.

This stepping procedure leads to two distinct phases. First there is a burn-in period in which Snake performs a greedy maximum-likelihood search to find the peak of the likelihood distribution with index \( i_{\text{peak}} \). Then, once the maximum has been located, the area around the peak is investigated by stepping to an unexplored neighbor of the surface point with the highest likelihood. This ensures that the surface grows outward according to the underlying likelihood distribution, and continues until all the surface points have reached the threshold. A large threshold value lets Snake investigate a larger parameter volume which ensures that the tails are investigated more closely; however, this requires more evaluations to be performed which is computationally expensive. The threshold should therefore be kept low enough to keep cost reasonable, but still large enough to make sure the edges are properly investigated.

The likelihood evaluation is by far the most time consuming component of cosmological evaluations, which implies that efficiency can be increased by parallelizing the Snake algorithm. As should be clear from the above, Snake is algorithmically trivial; it is nothing but an old-fashioned grid evaluation with efficient stepping and convergence. The only somewhat intricate part is to implement adequate book-keeping, which is necessary in order to maintain computational efficiency as the number of data elements, \( V \), in the repository increases. For this purpose, we implement dictionaries, based on the C++ standard map template. These maps store the combination of two values, the key and the mapped value, and enable access to the mapped value by using the corresponding key in constant time, as opposed to \( O(V) \) for unsorted lists or \( O(\log V) \) for sorted lists.

Two such maps are implemented in Snake. The first for keeping track of which point in parameter space corresponds to which iteration, which is used to check if the neighbors of the current point have already been visited, and if so returns their iteration index. The second map keeps track of the likelihood value corresponding to each iteration and is sorted according to descending likelihood such that the first point on the list will always be that with the highest likelihood. Therefore, the \( i_5 \) index is simply the mapped value at the top of the map. When a point becomes an interior point the corresponding entries are removed from the two map structures in order to keep these as short as possible and to avoid getting stuck at the overall maximum likelihood.

2.2. Walk-through of a Two-dimensional Example

Before testing the algorithm on realistic cases, it is useful to walk through it step-by-step for a simple case, to gain some intuition for its behavior. In this section, we therefore first consider the small two-dimensional example illustrated in Figure 1 and Table 1. The unknown distribution to be mapped is marked in Figure 1 by dashed lines, corresponding to \( 1\sigma \), \( 2\sigma \), and \( 3\sigma \) contours, and the threshold to be reached is defined as the \( 3\sigma \) contour.

First, we initialize the code at \((0,0)\), which in this case happened to lie slightly below and to the left of the
maximum-likelihood point. We evaluate the likelihood, and insert this point into the first row of the repository (Table 1). At this stage, the first four columns are finalized, the surface flag is set to true, and none of the neighbor indices (indicated by the ind array of length $2N_{pix}$) are set, indicating that no neighbors have been evaluated yet.

Second, as specified by the algorithm, we now find the surface point with the highest likelihood, which of course is the point just added. We select one of its neighbors, which in this case happened to be $(-1, 0)$. We evaluate its likelihood, and insert this new point into the second row of the repository. We update the neighbor indices of both this new point and the original point to point to each other’s main index. We then repeat this process over and over again, adding more and more points to the repository, until the smallest difference between the likelihood of the overall maximum-likelihood point and that of any point on the surface is larger than a predefined threshold.

Table 1 gives a snapshot of the repository (parameters, likelihood, current status of the ind array, and the surface flag) at iteration number 29, matching the illustration seen in Figure 1. The beige boxes correspond to the points in parameter space which lie on the surface, red boxes are interior points, and the blue box corresponds to the overall maximum likelihood. The green box is the parameter point on the surface with the highest likelihood and will be the start point for the next iteration. The numbers inside the boxes correspond to the iteration index, thus the path Snake takes to reach the maximum likelihood can be seen, as well as the relation between neighbors and the values of the first and last eight points quoted in the ind array in Table 1. Iterations which have all ind columns filled have their surface flag set to false and the point no longer exists in the maps. The process continues until all boxes touching the $3\sigma$ contour have turned red, after which the surface lies fully below the threshold.

### 2.3. Exploration of Double-peaked Likelihood

A second illustration of how Snake investigates parameter space is given by the double-peaked two-dimensional likelihood

$$L = A_1e^{\frac{1}{2}(x-\mu_1)^T C_1^{-1}(x-\mu_1)} + A_2e^{\frac{1}{2}(x-\mu_2)^T C_2^{-1}(x-\mu_2)},$$

where $x$ is the two-dimensional parameter vector, $A_1$ and $A_2$ are the peak amplitudes, $C_1$ and $C_2$ the corresponding covariance matrices, and $\mu_1$ and $\mu_2$ the vectors of the means.

The leftmost plot of Figure 2 shows a likelihood distribution that can be described by this equation for a particular set of
covariance matrices and means. The path Snake takes in the two-dimensional parameter space is shown in the rightmost plot of Figure 2 and as can be seen, Snake quickly finds the maximum likelihood of the closest peak, and then proceeds by investigating the area around this peak by visiting neighbors of the surface point with highest likelihood. When the likelihood being investigated falls to the value corresponding to the intersection of the two peaks Snake makes its way to the second peak, and continues by investigating the area around this peak in the same manner as the first peak. Once Snake returns to the likelihood equal to that at the intersection it will investigate the points around both peaks until the desired threshold is reached. 

Note that if the two peaks had been so far apart that the likelihood at the intersection fell below the threshold cutoff, the second peak would remain undiscovered. This problem can be solved in the same way as for standard Metropolis–Hasting samplers: run several Snakes in parallel with different initial positions. Once two independent Snakes touch for the first time, merge the repositories and the CPU working groups into one master–slave organization.

3. ACCURACY AND EFFICIENCY WITH INCREASING DIMENSIONALITY

The main outstanding question regarding the Snake algorithm is how well it scales with the number of dimensions in terms of efficiency. To study this question quantitatively, we consider a correlated Gaussian likelihood of the form

$$\mathcal{L} = e^{\frac{1}{2}(x-\mu)^T C^{-1}(x-\mu)},$$

where \(x\), \(C\), and \(\mu\) are the multidimensional parameter vector, covariance matrix, and vector of means, respectively. For simplicity both the mean and standard deviation for dimension number \(i\) are chosen to equal the value \(i\), thus \((\mu_i, \sigma_i) = (i, i)\).

Our goal is now to map out this distribution in \(N_{\text{par}}\) dimensions, and determine the maximum number of dimensions that can be probed with high accuracy using reasonable computational resources. To do so, we impose a limit on the number of likelihood evaluations of \(N = 10^6\), a typical number for modern cosmological analyses. The grid cell width in dimension \(i\) to be decreased arbitrarily, but since the cost faces an exponential growth with increasing \(N_{\text{par}}\), it seems reasonable to define the operational range for Snake to be \(N_{\text{par}} \leq 12–15\).

4. SEVEN-YEAR WMAP LIKELIHOOD ANALYSIS

4.1. Parameter Estimation

We now apply this method to the seven-year WMAP likelihood, and estimate cosmological parameters within the well-established six-parameter ΛCDM concordance model (Komatsu et al. 2011). The parameter set of choice is \(\Omega_b h^2\), \(\Omega_{DM} h^2\), \(\theta\),
Table 2. The agreement between the two methods is excellent, Figure 4, and means and standard deviations are tabulated in Snake and CosmoMC for comparison purposes.

| Parameter     | CosmoMC       | Snake         | Shift in $\sigma$ |
|---------------|---------------|---------------|-------------------|
| $\Omega_b h^2$ | 0.02252$^{+0.0055}_{-0.0056}$ | 0.02252$^{+0.0057}_{-0.0056}$ | 0 |
| $\Omega_{DM} h^2$ | 0.1110$^{+0.055}_{-0.054}$ | 0.1107$^{+0.055}_{-0.054}$ | 0.06 |
| $\theta$      | 1.039$^{+0.003}_{-0.002}$     | 1.039$^{+0.003}_{-0.002}$     | 0 |
| $\tau$        | 0.08849$^{+0.0032}_{-0.0034}$ | 0.08758$^{+0.0155}_{-0.01426}$ | 0.08 |
| $n_s$         | 0.9682$^{+0.0138}_{-0.0136}$  | 0.9681$^{+0.0139}_{-0.0138}$  | 0.07 |
| log$[10^{10} A_s]$ | 3.082$^{+0.034}_{-0.035}$    | 3.080$^{+0.035}_{-0.035}$    | 0.06 |

Notes. Comparison of best-fit parameters derived by CosmoMC and Snake from the seven-year WMAP data. $	au$, $n_s$, and log$[10^{10} A_s]$. The same setup is analyzed using both Snake and CosmoMC for comparison purposes.

The resulting normalized marginal distributions are shown in Figure 4, and means and standard deviations are tabulated in Table 2. The agreement between the two methods is excellent, with a maximum difference between the two methods corresponding to a 0.08$\sigma$ shift in $\tau$ and 0.07$\sigma$ shift in $n_s$.

The CosmoMC results were obtained with an MPI convergence criterion of 0.03, while the Snake convergence threshold was defined to be $-6.0$. Both codes were run on 50 CPUs, and the resulting wall times were 1.42 and 1.24 hr, respectively.

4.2. Model Selection by Bayesian Evidence

A significant advantage of Snake over CosmoMC is its direct access to the Bayesian evidence (e.g., Gelman et al. 2003). For a given model $H$ with parameters $\theta$ and data $d$, this is simply the normalization factor, $E \equiv P(d|H)$, in Bayes’ theorem,

$$P(\theta|d, H) = \frac{P(d|\theta, H)P(\theta|H)}{P(d|H)}.$$  

(3)

The other factors are the likelihood, $L(\theta|H) = P(d|\theta, H)$, the prior, $P(\theta|H)$, and the posterior, $P(\theta|d, H)$. Different models can be compared in terms of their evidence, which for a model, $H_n$, is given by

$$P(d|H_n) = \int_{\Omega} P(d, \theta|H_n) d\theta = \int_{\Omega} P(d|\theta, H_n)P(\theta|H_n) d\theta,$$  

(4)

where $P(d, \theta|H_n)$ is the joint probability distribution of $d$ and $\theta$ given this model over all of parameter space, $\Omega$, with step sizes of $d\theta$.

Calculating the evidence for different models using results from Snake is rather straightforward as the parameter space is gridded into even cells of volume $d\theta$. The integral in Equation (4) becomes a sum of the likelihood values within the threshold multiplied by the volume of one grid cell, where we assume a uniform prior which gives a factor of $1/L$ for each parameter, where $L$ is the range for each parameter.

To compare two different models, $H_1$ and $H_2$, it is common to consider the quantity

$$\delta \log E = \log E_1 - \log E_2,$$  

(5)

where $E_1$ and $E_2$ are the evidences of models $H_1$ and $H_2$, respectively. The larger the value of $\delta \log E$ the higher the evidence in favor of model $E_1$. To calibrate this quantity, one commonly adopts the Jeffreys’ scale (Liddle et al. 2006; Trotta 2008),

$$\delta \log E > \begin{cases} 
1 & \text{evidence for } E_1 \text{ is substantial} \\
2.5 & \text{evidence for } E_1 \text{ is strong} \\
5 & \text{evidence for } E_1 \text{ is decisive} 
\end{cases}.$$  

However, one should note that this scale only provides a general guideline, and conclusions can be application specific; see, e.g., Nesseris & Garcia-Bellido (2013) for a recent discussion of this issue.

We now evaluate the evidence for both the standard six-parameter model described above and for the reduced model obtained by enforcing $n_s = 1$. We find that the individual evidences are $E_1 = -3743.21$ and $E_2 = -3744.62$, respectively,
with an estimated uncertainty in each of 0.1. This corresponds to $\Delta \log E$ of 1.41 in favor of the six-parameter model; the full model therefore provides a better fit to the data, even when accounting for the larger parameter volume. Similar results have already been published by Parkinson & Liddle (2010).

As mentioned the evidence is dependent on the volume of the grid points investigated, which in turn is highly dependent on the preselected threshold. If the threshold is too low the volume covered by the analysis will be too small to compute the full evidence since the points excluded would have values high enough to contribute significantly to the evidence. In other words, one must select a sufficiently large threshold such that the values of excluded points would be insignificant, to ensure that the evidence has converged. Figure 5 shows the logarithm of the evidence as a function of threshold for the standard six-parameter model and for the reduced five-parameter model where we have imposed $n_s = 1$. As can be seen the log-evidence for the reduced model is fully converged at a threshold of $-6.0$ whereas that for the full model is very close to converged. Thus, a threshold value of $-6.0$ is large enough to ensure evidence convergence while still being low enough to keep the number of grid points as low as possible.

5. SUMMARY AND OUTLOOK

In this paper we have described a simple grid-based estimator for multi-dimensional likelihoods. This algorithm exploits the fact that by far most of the $N_{\text{par}}$-dimensional parameter volume in a general likelihood has negligible contributions, and spends its computational resources only where the likelihood itself is significant. However, in contrast to standard MCMC methods, it only considers each parameter point once, relying on the actual value of the likelihood.

The main advantages of this method are (1) trivial extraction of arbitrary conditional distributions; (2) direct access to Bayesian evidences; (3) better sampling of the tails of the distribution; and (4) nearly perfect parallelization scaling. The main disadvantage is a computational cost increasing exponentially with $N_{\text{par}}$. However, we have shown that the algorithm is fully capable of probing at least $N_{\text{par}} \lesssim 12–15$ with reasonable computational resources, which is sufficient for current cosmological models.

In the current implementation the total cost of the method is comparable to that of CosmoMC for similar convergence criteria. However, the cost for a full Snake analysis can be vastly reduced by introducing adaptive grids, in which the grid cell depends on the local properties of the likelihood, such that high-significance regions are sampled more densely than the tail regions. The results from this extension will be reported in a future publication.

The computations presented in this paper were carried out on Titan, a cluster owned and maintained by the University of Oslo and NOTUR. H.K.E. acknowledges support from the ERC Starting Grant StG2010-257080.

REFERENCES

Agarwal, S., Abdalla, F. B., Feldman, H. A., Lahav, O., & Thomas, S. A. 2012, MNRAS, 424, 1409
Auld, T., Bridges, M., Hobson, M. P., & Gull, S. F. 2007, MNRAS, 376, L11
Daniel, S. F., Connolly, A. J., & Schneider, J. 2012, arXiv:1205.2708
Fendt, W. A., & Wandelt, B. D. 2007, ApJ, 654, 2
Feroz, F., Hobson, M. P., & Bridges, M. 2009, MNRAS, 398, 1601
Frommert, M., Pflüger, D., Riller, T., et al. 2010, MNRAS, 406, 1177
Gelman, A., Carlin, J. B., Stern, H. S., & Rubin, D. B. 2003, Bayesian Data Analysis (2nd ed.; Chapman & Hall)
Hajian, A. 2007, PhRvD, 75, 083525
Jarosik, N., Bennett, C. L., Dunkley, J., et al. 2011, ApJS, 192, 14
Kilbinger, M., Benabed, K., Cappe, O., et al. 2011, ArXiv:1101.0950
Komatsu, E., Smith, K. M., Dunkley, J., et al. 2011, ApJS, 192, 18
Lewis, A., & Bridle, S. 2002, PhRvD, 66, 103511
Liddle, A., Mukherjee, P., & Parkinson, D. 2006, A&G, 47, 30
Nesseris, S., & Garcia-Bellido, J. 2013, JCAP, 08, 036
Parkinson, D., & Liddle, A. R. 2010, PhRvD, 82, 103533
Planck Collaboration 2011, A&A, 536, 1
Sandvik, H. B., Tegmark, M., Wang, X., & Zaldarriaga, M. 2004, PhRvD, 69, 063005
Trotta, R. 2008, ConPh, 49, 71
York, D. G., Adelman, J., Anderson, J. E., Jr., et al. 2000, AJ, 120, 1579