CosmoMC++

AL+SB
(Dated: August 23, 2006)

Discussion of proposal density, slice sampling and use of fast and slow parameters in CosmoMC.

I. PROPOSAL DENSITY

Metropolis will work with any symmetric proposal distribution in principle, as long as it eventually allows you to explore the full parameter space. A common test case for potential proposal distributions is sampling from a Gaussian distribution, which we consider here.

Frequently people choose an N-D Gaussian as the proposal distribution, e.g. Refs. [1, 2]. It pays to transform to uncorrelated parameters, and one can also re-scale so that parameters are centred on zero and have unit variance. Assume this has been done. In N-D the standard deviation of the proposal Gaussian should scale $\propto 1/\sqrt{N}$ for the rms distance of the proposal the same in all dimensions. Sampling from an N-D Gaussian then amounts to choosing a direction at random (a direction on the surface of an $N-1$ sphere), then making a proposal $P_N(r)$ for the distance r along that direction with

$$P_n(r) \propto r^{n-1} e^{-nr^2/2}.$$

The distance may be scaled by a factor s, and Ref. [2, 3] show that $s \sim 2.4$ is optimal in terms of minimizing the correlation length. Define $P_n^s(r) \equiv P_n(r/s)$. For $n > 1$ the proposal distribution goes to zero at $r = 0$, an hence encourages movement of the chain. One might think this is a Good Thing.

One simple improvement on this is to cycle directions rather than choosing a random direction for each move. There are N orthogonal vectors that make a basis for the space, but the basis can be oriented how you like. First choose a random basis, then cycle through the basis, making proposals along each basis vector direction. When all the basis vector directions have been proposed, generate a new random basis. This cycling avoids random directions sometimes heading back where they have just come from, and helps to remove random noise from the number of proposals in different directions. Experimentation shows that it does indeed improve performance a bit, with the effect being most noticeable for a smaller number of parameters.

Perhaps we could improve things by using a different proposal? Perhaps, but we need to be careful when assessing whether things are better or not. For example the autocorrelation function can be somewhat misleading. The plots in Fig. 1 show 1D chains using $P_n^{2.4}(r)$ as a proposal distribution for various n. As n increases the average accepted jump size increases. Calculating the correlation lengths, they are found to decrease dramatically with n. However as is clear from the plot, this is at the expense of a loss of efficient parameter space exploration: as $n \to \infty$ the chain is stuck on a grid of points with unit spacing, all other points are unobtainable. Therefore we want to be careful in using the correlation length to asses how good a proposal distribution is, as short correlation length does not necessarily imply good exploration (similar comments apply to using spectral efficiency estimates [2] which can also give misleading answers). In this case the Raftery and Lewis [4] (RL) statistics for percentile convergence show that increasing n actually make the distance between independent samples longer.

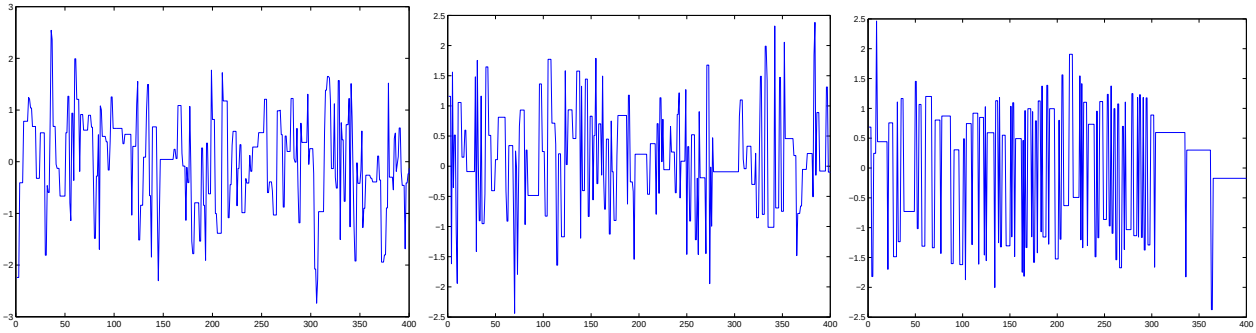

FIG. 1: Metropolis chain positions in a 1D Gaussian using proposal $P_n^{2.4}$, $n = 1, 2, 40$ (left to right). The correlation length decreases from left to right.

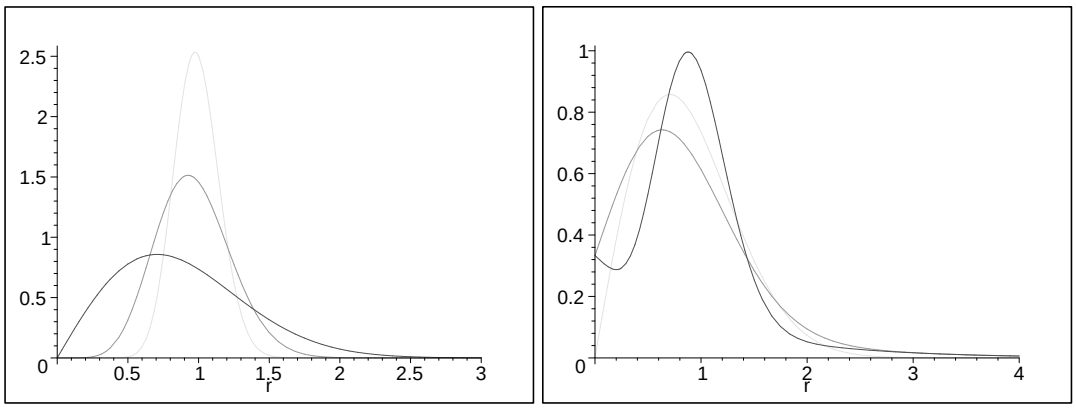

FIG. 2: Left: The distributions $P_n(r)$ for $n = 2, 7, 20$ (becoming more concentrated around 1). Right: The distribution $P_2(r)$ compared to broader tailed mixtures with an exponential ($f = 1/3$, $n = 2, 5$).

Consider separating the choice of direction to move in and distance to move. For example we could try some radial proposal function $P_n(r)$ for $n \neq N$ when sampling in N-D. As in the 1D case, the correlation length decreases as n of the proposal increases (though not very strongly), but the RL thinning factors get worse for $n > 2$. As $n \to \infty$ the delta function at $r = 1$ in a random direction corresponds to proposing on a thin $n-1$-shell centred on the current position, excluding all moves that are not close to distance 1 away (for large dimension N, an N-D Gaussian has almost all its probability around the unit $N-1$ shell). This is not necessarily bad as the randomizing of directions ameliorates the exploration problem. However the fact that RL factors get worse suggests that using large n may not be a good idea.

A good and robust proposal distribution performs well in the ideal case, but also doesn't perform too badly if the proposal width is not quite correct (for example if it has been estimated from a fairly short initial sampling period). In this respect $P_2^{2.4}$ is significantly superior to $P_N^{2.4}$ when sampling number of dimensions $N > 2$: the distribution is significantly broader, especially near zero, and therefore much less sensitive to the width being chosen incorrectly. For the case of a factor four overestimation of the proposal width, for 7D Gaussian sampling with $P_2^{2.4}$ the autocorrelation at 50 steps is $C_{50} \sim 0.45$, whereas for $P_7^{2.4}$ it is $C_{50} \sim 0.9$. For factor 4 underestimation $P_7^{2.4}$ performs only slightly better. For factor 8 overestimation C_{10} is close to one and $C_{50} \sim 0.8$ for $P_2^{2.4}$, which is slow but not completely useless; however $P_7^{2.4}$ performs catastrophically. For the ideal case when the proposal width is matched, $P_7^{2.4}$ gives $C_{10} \sim 0.3$ as opposed to $P_2^{2.4}$ which gives the slightly less good $C_{10} \sim 0.42$. However the Raftery and Lewis statistics for the latter case are actually slightly better, so it would seem that all-round using $P_2^{2.4}$ for the distance proposal in all dimensions $N > 1$ is likely to be a good idea, and certainly more robust than using an N-D Gaussian. Having a broader proposal distribution is also likely to be advantageous when the target is non-Gaussian, for example containing broad tails or nearly-isolated local maxima.

In summary we advocate choosing an orthonormal basis with random orientation, cycling in the basis vectors, and proposing moves a distance r in each basis vector direction with $P(r) = P_2^{2.4}(r)$. After a cycle is complete (or a small number of cycles), a new random basis is generated in a different random orientation. This is equivalent to sampling from a 2-D Gaussian in random (cycling) 2-D subspaces, which is close to what CosmoMC was originally doing [5] (which was using random 1D to 3D subspaces).

This can be made even more robust to wrong size estimation by increasing the probability near zero and at large numbers, which may also help with exploration of non-Gaussian shapes. A simple way to do this is do mix in some component of an exponential distribution:

$$P_{nf}(r) = fP_n(r) + (1-f)e^{-r}.$$

Taking $f \sim 2/3$, $n = 2$ seems to work fine, with performance not much effected by the choice. By design the efficiency is not very sensitive to the proposal width, 2.4 works fine. The distribution shape is rather similar to using Gaussian proposals in random 1 to 3D subspaces, but with somewhat broader tails at large r. An even more conservative option would be to add in some component of a Cauchy distribution ($\sim 1/(1+r^2)$, which has very long tails).

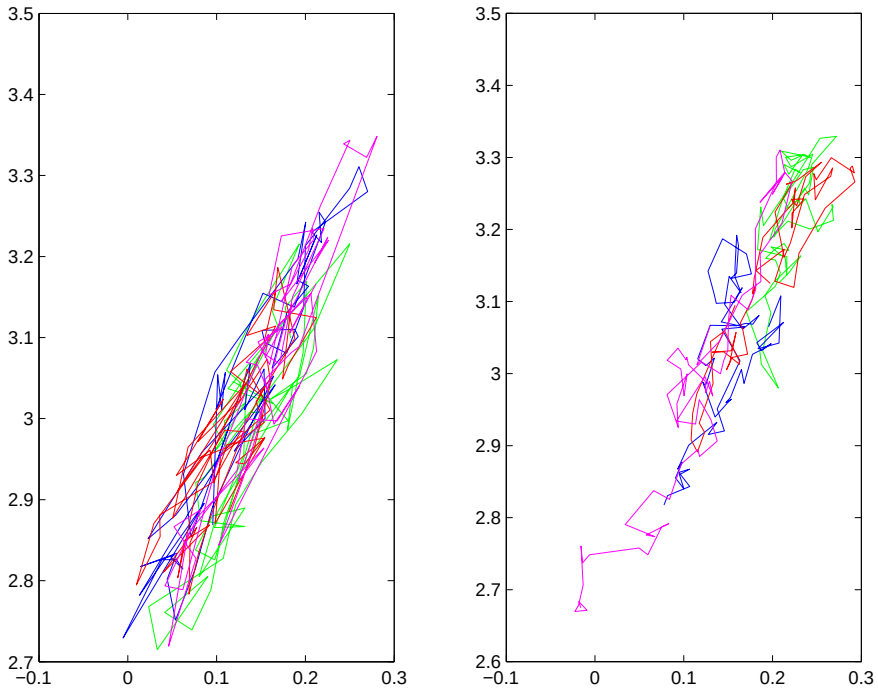

FIG. 3: Four chains sampling a 6D correlated Gaussian using cyclic 1D slice sampling without knowing the covariance matrix. Left randomizes the directions at the end of each cycle, right used a fixed basis in the original parameters. Both are thinned by 50.

Covariance estimation (estimate_propose_matrix = T)

Performance of Metropolis sampling depends strongly on how good the proposal distribution is and is poor if there are highly correlated parameters. An estimate of the covariance matrix can be used to transform to more orthogonal parameters. The covariance can be estimated from previous runs (or learned incrementally as in CosmoMC), however it is much more efficient if there is a way to get a good estimate easily before starting. The simplest way to do this is to compute the curvature matrix about the best fit point. The best fit point may also be of interest in itself, and is estimated by CosmoMC using a conjugate gradient algorithm (with numerical evaluation of derivatives). Convergence is deemed to have been reached when the log likelihood changes by only delta_loglike (set in the ini file) from one iteration to the next. Once an estimate of the best fit point is found, the second derivatives are estimated numerically. This seems to be a good way to proceed for well-behaved posterior distributions when previous runs are not available (for example when adding in new parameters, esp. if they are potentially highly correlated).

II. SLICE SAMPLING (SAMPLING_METHOD = 2)

Slice sampling [6, 7] is more robust than Metropolis methods as in simple cases it only performs linearly badly when the proposal density is not well matched to the target distribution. It may therefore be useful when using new parameters where little is known about the target distribution (and covariance estimation methods work badly).

Slice sampling is simplest in one dimension. In n-D one can simply cycle a set of directions that span the n-D space. If the distribution has diagonal correlations they have to be explored by random walk which is very inefficient. The efficiency of the slice sampler can be improved by randomizing the set of basis directions, as then sometimes one basis vector will point along the degenerate direction, and the stepping out rapidly samples along the entire diagonal direction. We therefore use a scheme like for the Metropolis case, where a random set of basis vector is generated, then slice sampling is performed along each in turn, then generating a new random set of basis vectors. If the distribution is uncorrelated this shouldn't lose anything, but (at least in low dimensions) will help significantly if it is.

When the distribution is understood, and there is a good Metropolis sampler to use, slice sampling requires more function evaluations and is slower. It may however be useful for a preliminary burn in period to assess the covariance, before continuing with Metropolis proposals using the estimated covariance matrix.

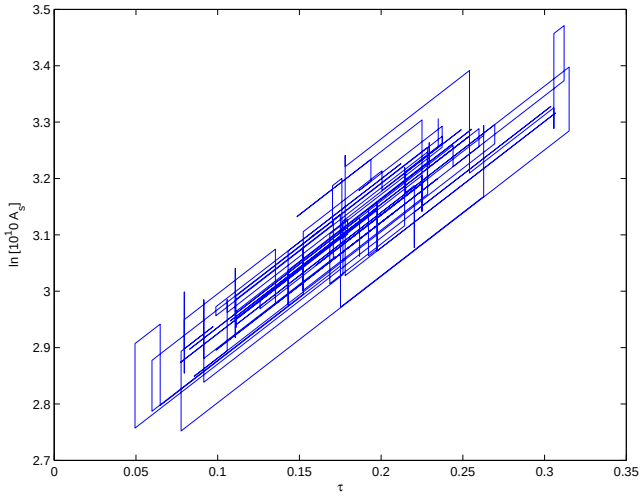

FIG. 4: 2D example sampling fast (vertical) and correlated slow (horizontal) variables.

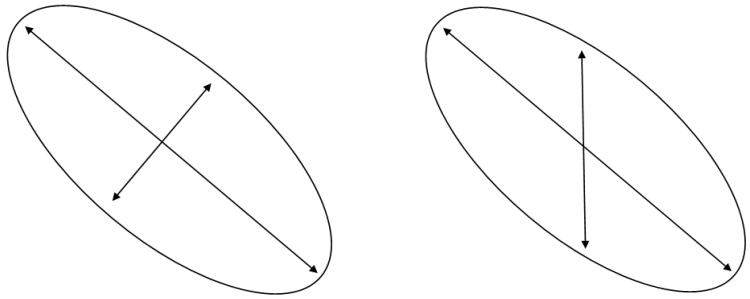

FIG. 5: Eigenvector choice, optimal fast/slow choice.

Slice sampling in the fast parameters combined with Metropolis in the optimal slow subspace may also sometimes be a good idea (sampling_method = 3).

If the stepping out procedure is omitted (safe if, for example, you set the initial step size to the width of the prior) function evaluations can be saved, but will be bad if the trial size is too small. If it is well matched, it might appear to be as good as Metropolis. However in n-D, a Gaussian Metropolis proposal that is accepted will move a distance close to 1, whereas the slice sampling is uniform in the slice, and hence makes moves close to zero just as likely as moves close to 1. So it may be less efficient for this reason. Or better if making moves close to 1 is actually bad (despite appearing good as far as reducing autocorrelations goes). Or it may be better because the slice can be larger than one. And slice sampling always moves at least a bit. However in the simple Gaussian test case slice sampling does appear to be worse than a well matched Metropolis.

III. FAST AND SLOW PARAMETERS

The idea is to use a known covariance matrix for the fast and slow parameters to change the slow parameters along eigendirections in which they change rapidly, and alternate with changing just the fast parameters in the fast subspace.

For a covariance matrix C, the covariance matrix of the fast parameters given a set of slow parameters is $(PC^{-1}P)^{-1}$ where P projects into the fast subspace. We explore the fast parameter space at a given value of the slow parameters by using eigendirections from this matrix.

For the slow parameters we want to move along eigendirections of the full covariance $C = UDU^T$ to maximize the acceptance probability and mobility (and hence minimize the total number of slow evaluations). We want to choose an $N_{\rm slow}$-dimensional subset of the e_i that span the slow parameter space. We take the ith eigenvector e_i, with the

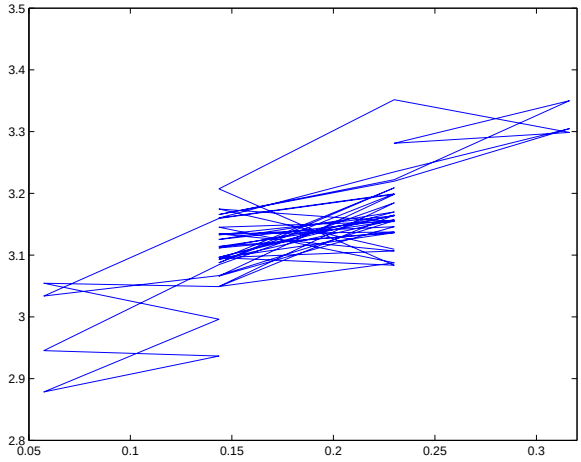

FIG. 6: Toy example of 222 chain steps in one directional grid sampling step sampling a 2D correlated Gaussian. The proposal distribution for the fast (vertical) and slow directions are parallel to the axes (covariance unknown). Only 6 slow function evaluations are required (the 4 points shown that are accepted, plus two rejected end points), but the chain has successfully traversed the entire space available on this grid.

standard deviation in that direction $\sigma_i = \sqrt{(D_{ii})}$ to make the vector $v_i = P'\sigma_i e_i$, which is the projection of one axis on an N-D ellipse into the slow subspace ($P' = 1 - P$). We choose the e_i so that v_i most efficiently moves around the slow space (in units of the slow parameter standard deviations). The first choice is simply the longest v_i. We then project the remaining vectors into the space orthogonal to v_i ($v'_j = v_j - v_j \cdot v_i v_i / v_i^2$), and find the next longest vector, etc.

In 2D (one fast, one slow) we simply make one diagonal move and one vertical (or horizontal) move. The diagonal move ensures fast movement in the slow parameter space. The orthogonal direction is explored using moves in the fast subspace, which is not orthogonal but nonetheless spans the full space. In this case using the above method saves a factor of two in computing time relative to moving in both eigendirections (assuming fast parameter time is negligible). Compared to moving in just the fast and slow subspaces separately the mobility is significantly improved by moving in the diagonal direction (much longer than the width parallel to the x-axis at any point), and hence significantly decreasing the number of slow evaluations needed to move to an independent point in parameter space. See Figs. 5, 4. This is the default (sampling_method =1).

Directional gridding (sampling_method = 4)

This is a method suggested by Radford Neal. Using the optimal slow subspace as described above, the idea is to make multiple proposals in each slow direction that is chosen, where each previous slow position is cached and fast parameters are also allowed to change. The idea is that this lets you explore a fast-slow degeneracy quickly, and allows the fast parameters to adapt to the changes in the slow parameters.

In detail, the steps are, for each random slow direction v in the cycle: 1. Choose a width w from some random distribution (e.g. $P_n^s(r)$); 2. Propose a move changing the current position by $\pm wv + f$, where the sign is chosen randomly and f is some random proposal for the fast parameters (if the current slow parameters have been computed before, get the results from the cache); 3. Accept the move according to the usual Metropolis rule; 4. Repeat steps 2/3 a number of times, then move on to the next slow direction v'.

This method can dramatically improve performance if there are fast-slow correlations and the covariance is not known, or there are local correlations not accounted for in the proposal distribution. See Fig. 6,7. It should also be more robust to underestimating the width of w as it can help convert a $\mathcal{O}(n^2)$ random walk to a $\mathcal{O}(n)$ stepping out (because the backward steps have been cached and so take near-zero time). For sampling a Gaussian distribution with an optimal proposal distribution the gridding method seems to perform somewhat worse than just making Metropolis moves in the optimal fast and slow subspaces, though not by much. In many cases this method may be a good idea.

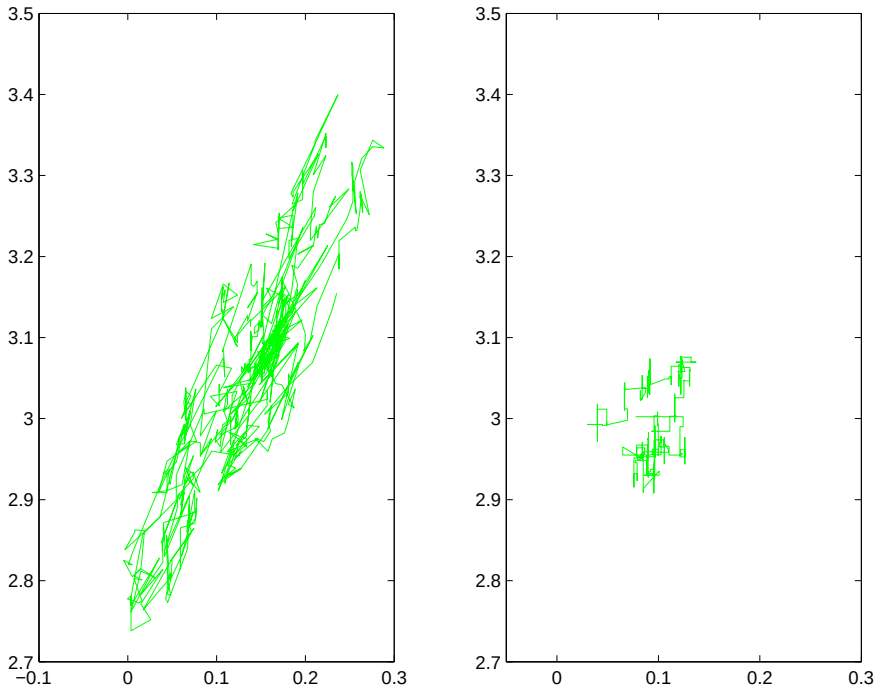

FIG. 7: Sampling a 5 slow + 2 fast dimensional correlated Gaussian without knowing the covariance matrix with alternating fast and slow Metropolis (Right) and using directional gridding (left) with 40 steps per slow direction. The chains are the length of 1500 slow evaluations. The fast proposal distribution was not chosen to well match the marginal width. (Actually this is a case where the restriction of the simple Metropolis proposals to the slow space is not a good idea - mixing in some random fast movement would allow occasional moves in the correct joint correlated direction; to this extent the comparison with simple Metropolis is unfair.)

Tempered transitions

More general schemes can be devised where one attempts to make transitions according to $P(\theta_{\rm slow})$ marginalized over the fast parameters $\theta_{\rm fast}$, which should be the optimal way to move about the slow parameter space. Radford Neal suggests various schemes where the marginalized probability is estimated by some means using lots of fast evaluations, but combined into one M-H move where the acceptance probability exactly corrects for the approximation. The one most likely to work when there are a large number of fast dimensions is based on tempered transitions [8]. The idea is to sample from $P(\theta_{\rm fast}|\theta_{\rm slow})^{1/T_i}$ for an increasing set of temperatures T_i from $T_i = 1$ to some large number T_m. At this high temperature propose in $P(\theta_{\rm slow}, \theta_{\rm fast})^{1/T_m}$, so the acceptance probability is close to one (if T_m is high enough). Then cool down in the new slow parameter position. The transition probability for the entire sequence of moves if essentially exactly that in Ref. [8], with a factor for the relative probabilities of the move at T_m. This procedure allows moves in the slow parameters to become nearly optimal, as well as allowing for arbitrarily complicated dependence of the likelihood on the fast and slow parameters. However experimentation indicates that of order 10000+ fast evaluations are required to make the acceptance probability remotely close to that expected from the marginalized probabilities, making the method only really useful for extremely fast parameters. I have therefore not implemented this in CosmoMC as most of the fast parameters are not fast enough: generally estimating $P(\theta_{\rm slow})$ is hard because it is the usual hard problem of working out the marginalized probability.

Radford Neal has written up a variation of the tempered transitions method by using a different interpolating distribution [9], however the number of fast evaluations required is still quite large, and it is not clear that it will work in all cases.

Flat histogram methods (sampling_method = 5,6)

These methods are aimed at sampling nastier distributions, for example multi-modal ones. sampling_method = 5 does multicanonical sampling [10, 11] and sampling_method = 6 does Wang-Landau sampling [12, 13] for a while,

then importance sampling from the estimated approximate inverse density of states. Both methods could be modified to calculate the evidence, but are only implemented for sampling from the target distribution at the moment. For nasty distributions they should perform better than simply doing normal MCMC at a high temperature, but typically are many times slower than standard sampling for simple distributions.

Nested Sampling

Nested sampling [14, 15] computes the evidence and generates weighted posterior samples. This is not in CosmoMC by default, but has been implemented for simple situations by others as an add-on [16]. Currently there is no public fully chain-based method[1] (the Parkinson implementation[2] uses an elliptical approximation).

IV. ACKNOWLEDGMENTS

We thank Radford Neal for many interesting and useful suggestions, and Jo Dunkley, Anze Slosar and Ben Wandelt for discussion.

[1] M. Tegmark et al. (SDSS), Astrophys. J. **606**, 702 (2004), astro-ph/0310725.
[2] J. Dunkley, M. Bucher, P. G. Ferreira, K. Moodley, and C. Skordis, Mon. Not. Roy. Astron. Soc. **356**, 925 (2005), astro-ph/0405462.
[3] A. Gelman, G. Roberts, and W. Gilks, in *Bayesian Statistics*, edited by J. M. Bernado et al. (OUP, 1996), vol. 5, p. 599.
[4] A. E. Raftery and S. M. Lewis, in *Bayesian Statistics*, edited by J. M. Bernado (OUP, 1992), p. 765.
[5] A. Lewis and S. Bridle, Phys. Rev. **D66**, 103511 (2002), astro-ph/0205436.
[6] R. M. Neal, Annals of Statistics **31**, 705 (2003), physics/0009028.
[7] D. J. C. MacKay (2002), `http://www.inference.phy.cam.ac.uk/mackay/itprnn/book.html`.
[8] R. M. Neal, Statistics and Computing **6**, 353 (1996), `http://www.cs.toronto.edu/~radford/ttrans.abstract.html`.
[9] R. M. Neal (2005), math.ST/0502099.
[10] B. A. Berg, Fields Inst. Commun. **26**, 1 (2000), cond-mat/9909236.
[11] J. Gubernatis and N. Hatano, Computing in Science & Engineering **2**, 95 (2002).
[12] F. Wang and D. P. Landau, Physical Review Letters **86**, 2050 (2001), cond-mat/0011174.
[13] F. Wang and D. P. Landau, Phys. Rev. E **64**, 056101 (2001), cond-mat/0107006.
[14] J. Skilling, in *American Institute of Physics Conference Series*, edited by R. Fischer, R. Preuss, and U. V. Toussaint (2004), pp. 395–405, URL `http://www.inference.phy.cam.ac.uk/bayesys/`.
[15] P. Murray, D. J. C. MacKay, Z. Ghahramani, and J. Skilling, Advances in Neural Information Processing Systems **18** (2006), URL `http://citeseer.ist.psu.edu/745739.html`.
[16] D. Parkinson, P. Mukherjee, and A. R. Liddle, Phys. Rev. **D73**, 123523 (2006), astro-ph/0605003.

[1] `http://cosmocoffee.info/viewtopic.php?t=586`
[2] `http://cosmonest.org`