A note on slice sampling

Research · June 2015

DOI: 10.13140/RG.2.1.1028.7207

2 authors:

Alessandro Rossi
European Commission
29 PUBLICATIONS **517** CITATIONS
SEE PROFILE

Christophe Planas
European Commission
31 PUBLICATIONS **536** CITATIONS
SEE PROFILE

MACFINROBODS – 612796 – FP7-SSH-2013-2

D8.1 – New approaches to improve Bayesian estimation methods using Gibbs sampling: software tools

Project acronym: *MACFINROBODS*

Project full title: *Integrated Macro-Financial Modelling for Robust Policy Design*

Grant agreement no.: 612796

Due-Date:	30 April 2015
Delivery:	29 April 2015
Lead Beneficiary:	JRC
Dissemination Level:	PU
Status:	submitted
Total number of pages:	23

This project has received funding from the European Union's Seventh Framework Programme (FP7) for research, technological development and demonstration under grant agreement number 612796

A note on slice sampling*

Christophe Planas and Alessandro Rossi

Joint Research Centre of the European Commission

April 2015

Abstract

We review some fundamental characteristics of the slice sampler algorithm proposed by Neal (2003). We first asses the importance of the scaling parameters and the procedures used to approximate the slice using a set of univariate distributions with differen features. Then we focus our attention on some multivariate versions of the slice sampler and we put forward a simple algorithm that works extremely well for high correlated variables. The performance of all samplers is measured in terms of efficiency and speed by means of several examples.

JEL CODE: C11, C15.

KEYWORDS: Gibbs Samplig, Monte Carlo Markov Chain, Multivariate sampling.

*This work is supported by the EU 7^{th} framework collaborative project Integrated Macro-Financial Modelling for Robust Policy Design (MACFINROBODS), grant no. 612796

1 Introduction

This note describes in some details a class of algorithms denominated slice samplers that arises in Bayesian statistics for sampling from the posterior distribution of model parameters. We first explains the main features of the slice sampler illustrating several alternative proposed by Neal (2003) both in a univariate and multivariate context. The performance of several slice samplers described here is assessed by means of an extensive Monte Carlo exercise based on several different test cases. For each test case we take the random walk Metropolis-Hastings algorithm (see e.g. Chib and E. Greenberg (1995)) as a benchmark.

2 Theory and practice of slice sampling

The slice sampler is a method for sampling from a generally unknown unnormilised continuous probability density function, say $f(\theta)$. It is a special case of the class of auxiliary variables methods (see e.g. Roberts and Rosenthal (1999)) that uses a single auxiliary variable. It is based on the following principle: a random variable γ is introduced to build the joint distribution of θ and γ, i.e. $p(\theta, \gamma)$, by taking the marginal distribution $p(\theta)$ unchanged. Writing $p(\theta) = f(\theta)/k$, $k = \int f(\theta) d\theta$, and choosing $p(\gamma \,|\, \theta)$ uniform over the set $(0, f(\theta))$ yields the joint density:

$$p(\theta, \gamma) = p(\gamma \,|\, \theta) p(\theta) = \frac{1}{f(\theta)} I_{[0<\gamma<f(\theta)]} p(\theta) = \frac{1}{k} I_{[0<\gamma<f(\theta)]}$$

Apart from trivial cases the expression above cannot be used directly to sample from the joint distribution of θ and γ. However given its structure it is suitable for implementation of a Gibbs sampling strategy which draws iteratively $\gamma \mid \theta$ and $\theta \,|\, \gamma$:

(i) $\gamma \,|\, \theta$ from a uniform distribution over the set $(0, f(\theta))$;

(ii) $\theta \,|\, \gamma$ from a uniform distribution over the set $S = \{\theta : \gamma < f(\theta)\}$.

Under mild conditions (Geman and Geman, 1984), the Markov chain generated by this sampling scheme has $p(\gamma, \theta)$ as its unique stationary distribution. Hence, inference about θ can be made simply looking at the marginal chain. Mira and Tierney (2002) have studied the rate of convergence of the slice simpler concluding that this sampling method has robust ergodic properties and hence it is an appealing algorithm from the theoretical point of view.

2.1 Approximating the slice

In general sampling directly from the slice S is not feasible. The reason is that the nature of the slice is generally unknown. Moreover the practice of drawing blindly θ from a uniform

distribution until a variate fall inside the slice can be highly inefficient. Thus we must resort to some clever way of implementing step (ii). In a seminal paper Neal (2003) puts forward several ideas on how to build a Markov chain that leaves the marginal distribution of θ invariant that differ in the way step (ii) is implemented. Any of those strategies share the following idea:

- position an interval $I = (L, R)$ around θ^0 at random that contains the slice S as much as possible;
- draw θ from the set $A = \{\theta : \theta \in S \cap I \;\; and \;\; \Pr(I \,|\, \theta) = \Pr(I \,|\, \theta^o)\}$

For correctness of the sampler is crucial that the set of acceptable successors A is built satisfying the detail balance condition: $\Pr(I \,|\, \theta) = \Pr(I \,|\, \theta^o)$. This condition states that the probability of choosing I given a generic element of A must be same of that if we were to choose I starting from θ^o. In the sequel we report three main strategies described in Neal (2013) to draw a new point, say θ^n, from S. To introduce matters we start with the univariate case.

Random positioning

Denote by u a variate from a uniform distribution over the unitary interval $(0, 1)$. Given a starting point θ^o, draw γ from $U(0, f(\theta^o)$. This defines the slice $S = \{\theta : \gamma < f(\theta^o)\}$. A first simple attempt to sample θ from S is as follows:

- position an interval $I = (L, R)$ around θ^o at random: i.e. set $L = \theta^o - uW$, and $R = L + W$;
- draw a candidate θ^c from a uniform distribution over the set I, i.e. set $\theta^c = L + u(R - L)$. Repeat until $\gamma < f(\theta^c)$, then set $\theta^n = \theta^c$.

This procedure may suffer of two main problems: the initial interval I could be too small or too large depending on the choice of the tuning parameter W. This is choice is crucial since if W is too small the next point θ^n will be close to the initial point θ^o generating autocorrelation in the corresponding Markov chain. On the contrary if W is too large the probabilty of the set defined by the intersection between S and I, i.e. $\Pr(\theta : \theta \in S \cap I)$ will be small, hence getting a draw within it will be hard. Attempts to solve both issues are considered in the next two procedures.

Stepping out

A reduction of the impact of the scaling parameters W on the efficiency of the sampler can be achieved enlarging the interval I when the latter turns out to be too small:

- position $I = (L, R)$ around θ^o at random: i.e. set $L = \theta^o - uW$, and $R = L + W$. Then expand I setting $L = L - W$ and $R = R + W$, until $\gamma < f(L)$ and $\gamma < f(R)$.

This ensures the I might contain at leat a non negligible part of the slice S or even the entire slice in case of unimodal distributions. It might be the case that at the end of the step above I turns out to be too large compared with S. In this case non accepted draws are used to shrink I:

- draw a candidate θ^c from a uniform distribution over the set I, i.e. set $\theta^c = L + u(R - L)$.

$$\text{set} \quad \begin{cases} L = \theta^c & if \quad \theta^n < \theta^o \\ R = \theta^c & otherwise \end{cases}$$

  repeat until $\gamma < f(\theta^c)$, then set $\theta^n = \theta^c$.

Doubling

A slightly different procedure is provided by Neal (2013) for approximating the initial slice. This approach is claimed to be faster in expanding the interval I than stepping out when W turns out to be too small.

- Position $I = (L, R)$ around θ^o at random: i.e. set $L = \theta^o - uW$, and $R = L + W$. Then expand I by setting $L = L - (R - L)$ if $u < 1/2$, and $R = R + (R - L)$ otherwise. Repeat until $\gamma < f(L)$ or $\gamma < f(R)$.

As can be seen the interval expand faster, furthermore sometimes we do not need to check both edges of the interval. The shrinking procedures is slightly different to ensure the validity of the detailed balance condition (see Neal 2013, Fig 6 for more details).

2.2 The performance of the univariate slice sampler

In this section we measure the performance of the slice sampler with respect to the choice of the scaling parameters W and the procedure which best approximate the slice among those described above: random positioning, stepping out, and doubling. For W we choose 4 values (after experimenting) that are multiple of the standard deviation of distribution under study. The reason is that albeit we do not know the scale of the target distribution before sampling, a rough estimates of it can be obtained in a transitory phase of the MCMC procedure. This exercise is interesting per se because a univariate version of the slice sampler cames out in practice applying a Gibbs scheme that draws a vector of parameters one-at-a-time from the full conditional distributions: $f(\theta_i \mid \theta_1, \cdots, \theta_{i-1}, \theta_{i+1} \cdots, \theta_d)$, $i = 1, 2, \cdots, d$.

As a benchmark sampler we take a plain version of the random walk Metropolis Hastings (RW-MH) whose proposal distribution is Gaussian with mean equal to $E(\theta)$ and standard deviation calibrated to get an optimal acceptance rate close to 0.25 as suggested by Gelman, Gilks, and Roberts (1997). The RW-MH is likely the most used MCMC sampler in real applications due to its simplicity and speediness.

As target densities we employ 12 out of 15 distributions described by Marron and Wand (1992). These are univariate mixture of Normals that show a variety of characteristics: asymmetry, lepto- or plato-kurtosis, multi-modality etc. The shape of each distribution is shown in Figure 1.

[*Insert Figure 1 here*]

For any version of the slice sampler described above likewise for the RW-MH algorithm we generate 500 samples of dimension $G = 10000$ with different initial conditions from the target distribution. For each sample $(\theta^1, \theta^2, \cdots, \theta^G)$ we measure the inefficiency factor (IF) and record the number of calls to $f(\theta)$. The inefficiency factor is often employed for evaluating the accuracy of sampling-based approaches to the calculation of posterior moments (see e.g. Geweke (1999)) since it provides an indication on the sample size G to get the desired accuracy in computing such moments. We measure inefficiency as follows:

$$IF = 1 + 2\sum_{j=1}^{p} \omega_j \rho_j$$

where ρ_j is the lag-j correlation of the sample $(\theta^1, \theta^2, \cdots, \theta^G)$, $p = 1000$, and ω_j are the Parzen-weights. The (in)efficiency of a sampler is linked to autocorrelation structure of the MCMC transition kernel that generates the draws: when $\rho_j = 0$, for any j, the sampler achieve the same efficiency of that of a sampler that generates independent draws. Viceversa high and slow decaying values of the autocorrelation implies lower efficiency.

The second aspect of interest is the number of calls to the posterior density or full conditional $f(\theta)$ to generate a single draw. This is because in many real applications the evaluation of $f(\theta)$ is time consuming. In a Dynamic Stochastic General Equilibrium (DSGE) model, for instance, a draw from the posterior of model parameters requires to solve the model fist, and to compute the likelihood function by the Kalman filter. The full process may requires seconds in medium-large scale models.

These two indicators are put together to compue the relative efficiency (RE) of a given sampler, say A, with respect to the efficiency achieved by the RW-MH. Relative efficiency is given by the ratio:

$$RE = \frac{IF_A \times Eval_A}{IF_{MH} \times Eval_{MH}}$$

To asses the effective ability of the various sampling-based approaches to converge to the true target distribution we count the number of rejections of the Cramer-von Mises (CvM) test (Csorgo and Faraway (1996)) at 5% level (not reported here) that compares the draws from the MCMC with the theoretical distribution. A given sampler produces acceptable results when the null of equal distribution is rejected in 5% of the replications (i.e. roughly 25 out of 500). We do not employ any convergence criterion statistics since convergence here is ensured by the CvM test. Results of the Monte Carlo experiment for the twelve mixture of Normals are reported in Table 1. The numbers reported in the table are the sample averages over the 500 replications.

[*Insert Table 1 here*]

Several facts deserves to be stressed: (i) the RW-MH algorithm exhibits IFs larger that those of slice algorithm almost uniformly over the test cases when employing the stepping out or the doubling procedure; (ii) for the slice sampler the symmetric distributions, no matter their degree of kurtosis, attain almost optimal efficacy with $IF \simeq 1$. On the contrary asymmetric distributions displays a lower efficiency with the "Strongly skewed" case showing the highest IFs almost uniformly over the samplers; (iii) to greater values of W correspond lower inefficiency factors uniformly over both the test cases and the procedures. In fact when W is large the slice sampler has higher mixing properties meaning that the probability of sampling a value of $theta$, say $theta^n$, which is far away from the previous draw $theta^{n-1}$ increases. On the contrary the number of evaluations to $f(\theta)$, as a function of W, is mainly U-shaped for the stepping out and doubling procedures, while it is monotonically increasing when the random positioning scheme is used. There is a clear trade-off between higher efficiency in terms of IFs and the number of evaluations. We cannot expect to minimize both at the same time; (iv) relative efficiency (RE) of the slice sampling with stepping out outperforms that of the RW-MH algorithm when W is in the range $[3, 10]\sigma$. Remarkably, the stepping out scheme requires on average no more that 6 evaluations to $f(theta)$ no matter the shape of the target distribution.

From this rather extensive exercise we claim that a slice sampler that implements the stepping out procedure using a scaling parameters $W \in [3, 10]\sigma$ delivers always samples with moderate autocorrelations after having evaluated the target density a fair number of times ($\simeq 6$). Yet the fact that results are loosely sensible to W poses this algorithm in the small class of samplers that requires a rather small amount of tuning.

3 Slice sampling for multivariate distributions

In this section we first review methods proposed in the literature for sampling from multivariate target densities using the principle of slice sampling. Then we propose a new algorithm that performs extremely well when a strong linear dependence among variables is present.

Suppose we need to sample from a probability density function $f(\theta)$ where θ is d-dimensional. The random positioning and doubling procedures outlined in the previous section do generalize to the multivariate case but are in general rather inefficient in the sense that generate sample draws with relatively high autocorrelation that increases quickly with for increasing values of d (results are not reported here). Conversely, the stepping out procedure works better in terms of inefficiency factor but demands for a very high number of evaluations of $f(\theta)$. In fact it requires the computation of 2^d vertices of the hypercube that approximates the slice. This problem can be partially offset using parallelization techniques as in Tibbits, Haran, and Liechty (2011).

In view of the above considerations and the results obtained in Section 2 we relay on multivariate slice samplers that implements the stepping out scheme setting $W_i = 3\sigma_i$, with $\sigma_i \equiv Var(\theta_i)^{1/2}$, $i = 1, 2, \cdots, d$.

3.1 A plain multivariate slice sampler

A generalization of the stepping out procedure for d-dimensional problems is obtained by replacing the scalar interval I introduced in section 2 with a d-dimensional axis-aligned hypercube $H = (V_1, \cdots, V_{2^d})$ whose vertices V_j, are defined trough a set of d-dimensional lower bonds $L = (L_1, L_2 \cdots, L_d)$, and upper bounds $R = (R_1, R_2 \cdots, R_d)$. We start from a given vector θ^o, after we draw $\gamma \sim U(0, f(\theta^o)$, that defines the slice $S = \{\theta : \gamma < f(\theta^o)\}$. Then we proceed as follows:

- position the axis-aligned hypercube $H = (V_1, \cdots, V_{2^d})$ around θ^o at random, i.e. set $L_i = \theta_i^o - uW_i$, and $R_i = L_i + W_i$, $i = 1, 2, \cdots, d$. Unless all the vertices of H are outside the slice S, i.e. $\gamma > f(V_j)$ for any j, expand H setting $L_i = L_i - W_i$ and $R_i = R_i + W_i$ for any i, which gives a new set of vertices $V_1, V_2, \cdots, V_{2^d}$;

- draw a candidate θ^c from a uniform distribution over the set H, i.e. set $\theta_i^c = L_i + u(R_i - L_i)$ for all i. Finally, along each dimension:

$$\text{set} \quad \begin{cases} L_i = \theta_i^c & if \quad \theta_i^n < \theta_i^o \\ R_i = \theta_i^c & otherwise \end{cases}$$

  repeat until $\gamma < f(\theta^c)$, then set $\theta^n = \theta^c$.

Note that this strategy becomes unfeasible for very large d. For moderately large values of d, say $d < 20$, the algorithm can be parallelized since the computation of the density evaluated in correspondence of any of the 2^d vertex have nothing in common.

3.2 Using the gradient to shrink the hypercube more efficiently

Shrinking the hypercubes in all directions any time a draw is rejected can be inefficient in the sense that the procedure is likely to stop with a new draw which is close to the the starting point. This has the unwanted effect of generating a Markov chain with high serial autocorrelation. We provide evidence on this later on. Neal (2013) suggests to use the gradient of $f(\cdot)$ evaluated at θ^c, $\frac{\partial f(\theta)}{\partial \theta}|_{\theta=\theta^c}$ any time this draw is rejected. In a bivariate context, his idea is sketched in Figure 7, on page 723.

3.3 A multivariate slice sampler with directional hypercubes

Here we propose a variant of the plain method to sample more efficiently from highly correlated variables. Axis-aligned hypercubes in these cases cover usually a small peace of the slice making the multivariate slice sampler, with or with out the use of gradient, highly inefficient. The idea is to rotate the hypercubes along the directions where much of the slice is concentrated.

We make use of the fact that the slice sampler is well suited for adaptation, i.e. previous draws can be used to extract useful information upon the joint posterior distribution (e.g. using draws obtained in the burn-in phase). Evidence of linear dependence between variables is contained in any estimate of the variance covariance matrix $\Sigma \equiv Var(\theta)$. A rough estimate of Σ can be obtained for instance by the autocorrelation-consistent estimator of Newey and West (1987) applied on the burn-in draws. Applying the spectral decomposition to Σ, we have $\Sigma = A\Lambda A'$, where the eigenvectors represented by the columns of A suggest the direction in which to rotate the axis, while the eigenvalues contained in the diagonal of Λ provide information about the length of the hypercube along any given direction.

The algorithm works as follows. Given θ^o, $\gamma \sim U(0, f(\theta^o)$, and $S = \{\theta : \gamma < f(\theta^o)\}$, then θ^n is obtained as follows:

- Set $W_i = 3\Lambda_{i,i}^{1/2}$, position $H = (V_1, \cdots, V_{2^d})$ around θ^o at random: $\tilde{L}_i = -uW_i$, and $\tilde{R}_i = \tilde{L}_i + W_i$, $i = 1, \cdots, d$, which gives $\tilde{H} = (\tilde{V}_1, \cdots, \tilde{V}_{2^d})$. Then H is obtained rotating the axis and centering on θ^o, i.e. $V_j = \theta^o + A\tilde{V}_j$.
- Expand H: set $\tilde{L} = \tilde{L} - W$ and $\tilde{R} = \tilde{R} + W$, which gives $\tilde{V}_j$. Then set $V_j = \theta^o + A\tilde{V}_j$ until $\gamma < f(V_j)$, $j = 1, \cdots, 2^d$.

- Shrinking H: draw $\tilde{\theta}^c = \tilde{L} + u(\tilde{R} - \tilde{L})$, and set $\theta^c = \theta^o + A\tilde{\theta}^c$. Finally

$$\text{set} \quad \begin{cases} \tilde{L}_i = \tilde{\theta}_i^{\,c} & if \quad \tilde{\theta}_i^c < 0 \\ \tilde{R}_i = \tilde{\theta}_i^c & otherwise \end{cases}$$

  repeat until $\gamma < f(\theta^c)$, then set $\theta^n = \theta^c$.

The new algorithm is similar to the plain one, and importantly it does not involve any additional evaluation of $f(\theta)$. However, in case of highly dependent variables, its performance can be much better than the pain one as will be shown in the sequel.

3.4 The performance of the multivariate slice sampling

We test the accuracy and efficiency of the algorithms presented above using three different examples. As before the performance of the samplers is assessed by a Monte Carlo experiment which generates 500 samples of dimension $G = 10000$ with different initial conditions from the target distribution. We make use of the same indicators described in section 2.4 for gauging efficiency.

The correctness of the multivariate samplers is assessed using a multivariate CvM statistics (Cotterill and Csorgo, 1982) in a way similar to that explained in Section 2. Results on the Cvm test are not reported here.

The algorithms under scrutiny are: the multivariate RW-MH, a Gibbs sampling scheme where the elements of the d-dimensional vector θ are sampled one-at-a-time using the univariate slice sampler with stepping out procedure and $W_i = 3\sigma_i$, the plain multivariate slice described in subsection 3.1, the multivariate slice sampler which make use of the gradient of $f(\theta)$ as described in subsection 3.2, and the multivariate version of the slice sampler developed in this note which rotates the hypercubes using information on the covariance structure of θ which is detailed in subsection 3.3.

Example 1: uncorrelated variables

Assume θ is a d-dimensional random vector with probability density function $N(0, S\lambda S)$, $S = diag(1, 5, \cdots, 5 * d)$, $\lambda = I_d$, and $d = 2, 5, 10$. The variables in θ are thus independent with a different scale. Results on IF, number of evaluations, and RE are reported in Table 2.

[*Insert Table 2 here*]

As expected both the IF's and RE's increase with the dimension of θ (d) for all algorithms but the one-at-a-time univariate slice sampler. Worth noting that the multivariate plain slice

sampler (with and without the use of the gradient) always outperforms the RW-MH in terms of IF. However to beat the RW-MH in terms of RE in a ten-dimensional context for instance we need to decrease CPU time of a factor of 300 for the plain and 170 when the gradient is used. This looks a quite demanding exercise even when using up-to-date parallelization techniques.

Example 2: highly correlated variables

Assume θ is a d-dimensional random vector with probability density function $N(0, S\lambda S)$, $S = diag(1, 5, \cdots, 5 * d)$, $\lambda = .95\mathbf{1}\mathbf{1}' + .05I_d$, $\mathbf{1}$ is the d-dimensional column vector made up of ones, and $d = 2, 5, 10$. The random vector is now made up of strongly linearly dependent variables with a different scale.

[*Insert Table 3 here*]

Three main considerations arise from Table 2: (i) the RW-MH is highly inefficient. The first-order autocorrelation of the chain is close to one so to question proper convergence to the stationary distribution. (ii) The one-at-a-time univariate slice sampler show increasing IF's as expected but the degree of inefficiency is still acceptable. (iii) The multivariate slice sampler with directional hypercubes behaves extremely well: it exhibits virtually no correlation for $d \leq 5$ and a very small degree of autocorrelation with $d = 10$.

Example 3: mixture of normals

Assume $\theta = (\theta_1, \theta_2)$ is a mixture of three bivariate normal distributions as in Gilks, Roberts, and Sahu (1998):

$$f(\theta) = \sum_{j=1}^{3} \omega_j \phi(\theta; \mu_j, \Sigma_j)$$

where $\phi(\cdot; \mu, \Sigma)$ is the bivariate normal density distribution with mean μ and covariance Σ, $\omega_j = 1/3$, $j = 1, 2, 3$, $\mu_1 = (0, 0)$, $\mu_2 = (-3, -3)$, $\mu_3 = (2, 2)$, $\Sigma_1 = I_2$, $\Sigma_2 = 0.9\ \mathbf{1}\mathbf{1}' + 0.1\ I_2$, and $\Sigma_3 = -0.9\ \mathbf{1}\mathbf{1}' + 1.1\ I_2$. The shape of this distribution is sketched in the upper panels of Figure 2 where the tree components of the mixture are clearly identifiable.

[*Insert Figure 2 here*]

Albeit the presence of non-linearities, the unconditional covariance matrix reveals a strong linear dependence between the two variables θ_1 and θ_2, which is around 0.8. This justifies the use of the slice sampler with directional hypercubes. Table 4 shows the results of the Monte carlo exercise.

[*Insert Table 4 here*]

Again the RW-MH works poorly with an inefficiency factor larger than 200. All the slice sampler algorithms tested here work much better with IFs almost 100 times lower than the one of the RW-MH. Remarkably, the multivariate slice sampler with directional hypercubes is by large the best sampler. To have and idea of the autocorrelation structure generated by both the RW-MH and the multivariate slice sampler with directional hypercubes the lower panels of Figure 2 shows a single run of 10000 draws obtained with these two samplers for θ_1. The draws of RW-MH are very persistent: the Markov chain generated by this transition kernel tends to remain in a given region for hundreds of runs. Conversely the chain generated by multivariate slice sampler with directional hypercubes shows a high degree of mixing.

4 Conclusions and future work

In this work we have tested the efficiency of several versions of the slice sampler put forward by Neal (2003). Using a battery of univariate test cases we have derived a rule of thumb for the scale parameter W, i.e. $W \in (3, 10)\sigma$. The results obtained demonstrated that the slice sampling loosely relays on tuning parameters. Furthermore we can claim that the stepping out scheme generally outperforms the doubling and random position ones. In a multivariate context we have seen that one-at-a-time Gibbs sampling that implements the univariate slice sampler works fine when the degree of linear dependence is not very high. In such cases the multivariate slice sampling with directional hypercubes is decisively a better alternative.

Next issues that we would like to address are: (i) the parallelization of the multivariate slice algorithm using Open MP, and/or CUDA on Graphical Processing Units (see for instance Tibbits et al. 2011); (ii) assessing the performance of the various slice sampler on small/medium-scale DSGE models; (iii) enlarge the set of samplers to include for instance the adaptive MH sampler (Haario, Saksman, and Tamminen, 2001), the MH sampler with randomized block (Chib and Ramamurthy, 2011), and the adaptive, independent proposal, MH sampler by fast estimation of mixtures of Normals (Giordani and Kohn, 2010).

References

D.S. Cotterill, and M. Csorgo (1982), 'On the limiting distribution of and critical values for the multivariate Cramer-von Mises statistic', *The Annals of Statistics*, 10, 1, 233-244.

S. Csorgo, and J. Faraway (1996), 'The exact and asymptotic distribution of Cramer-von Mises statistics', *Journal of the Royal Statistical Society B*, 58, 1, 221-234.

S. Chib and E. Greenberg (1995), 'Understanding the Metropolis-Hastings Algorithm', *The American Statistician*, 49, 327-335.

S. Chib and S. Ramamurthy (2010), 'Tailored randomized block MCMC methods with application to DSGE models', *Journal of Econometrics*, 155, 19-38.

Fuller (1996), 'Introduction to Statistical Time Series', *Wiley Series in Probability and Statistics*.

A. Gelman, W. R. Gilks, and G. O. Roberts (1997), 'Weak convergence and optimal scaling of random walk Metropolis algorithms', *Ann. Appl. Probab.*, 7, 1, 110-120.

S. Geman, and D. Geman (1997), 'Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images', *Pattern Analysis and Machine Intelligence, IEEE Transactions*, 6, 6, 721-741.

J. Geweke (1999), 'Using simulation methods for Bayesian econometric models: inference, development, and communication', *Econometric Reviews*, 18, 1, 1-73.

W. R. Gilks, G. O. Roberts, and S. K. Sahu (1998), 'Adaptive Markov Chain Monte Carlo through regeneration', *Journal of the American Statistical Association*, 93, 443, 1045-1054.

P. Giordani, and R. Kohn (2010), 'Adaptive Independent MetropolisHastings by Fast Estimation of Mixtures of Normals', *Journal of Computational and Graphical Statistics*, 19, 2, 243259.

H. Haario, E. Saksman, and J. Tamminen (2001), 'An adaptive Metropolis algorithm', *Bernoulli*, 7, 2, 223-242.

J. S. Marron and M.P. Wand (1992), 'Exact integrated squared error', *The Annals of Statistics*, 20, 2, 712-736.

A. Mira and L. Tierney (2002), 'Efficiency and convergence properties of slice samplers', *Scandinavian Journal of Statistics*, 29, 1, 1-12.

R. M. Neal, (2003), 'Slice sampling', *The Annals of Statistics*, 31, 3, 705-767.

W. K. Newey, and K. D. West (1987), 'A simple positive semi-definite, heteroskedasticity and autocorrelation consistent covariance matrix', *Econometrica*, 55, 703-708.

G.O. Roberts, and J. Rosenthal (2009), 'Convergence of slice sampler Markov chains', *Journal of the Royal Statistical Society B*, 31, 643-660.

M. M. Tibbits, M. Haran, and J. C. Liechty (2011), 'Parallel multivariate slice sampling', *Stat. Comput.*, 21, 3, 415-430.

Figures and tables

FIGURE 1 Marron and Wand (1992) densities

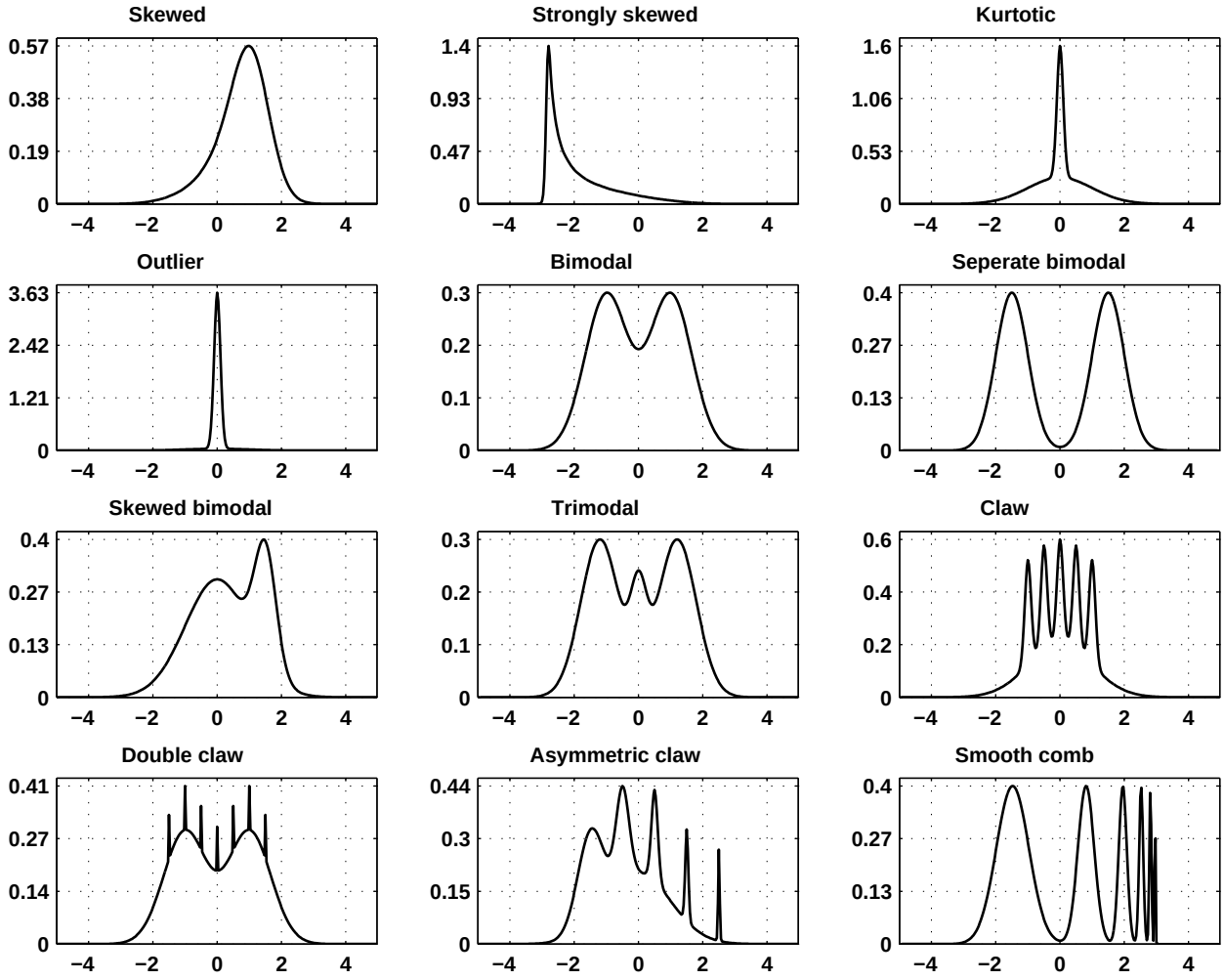

FIGURE 2 MCMC draws for the mixture of normals of example 3

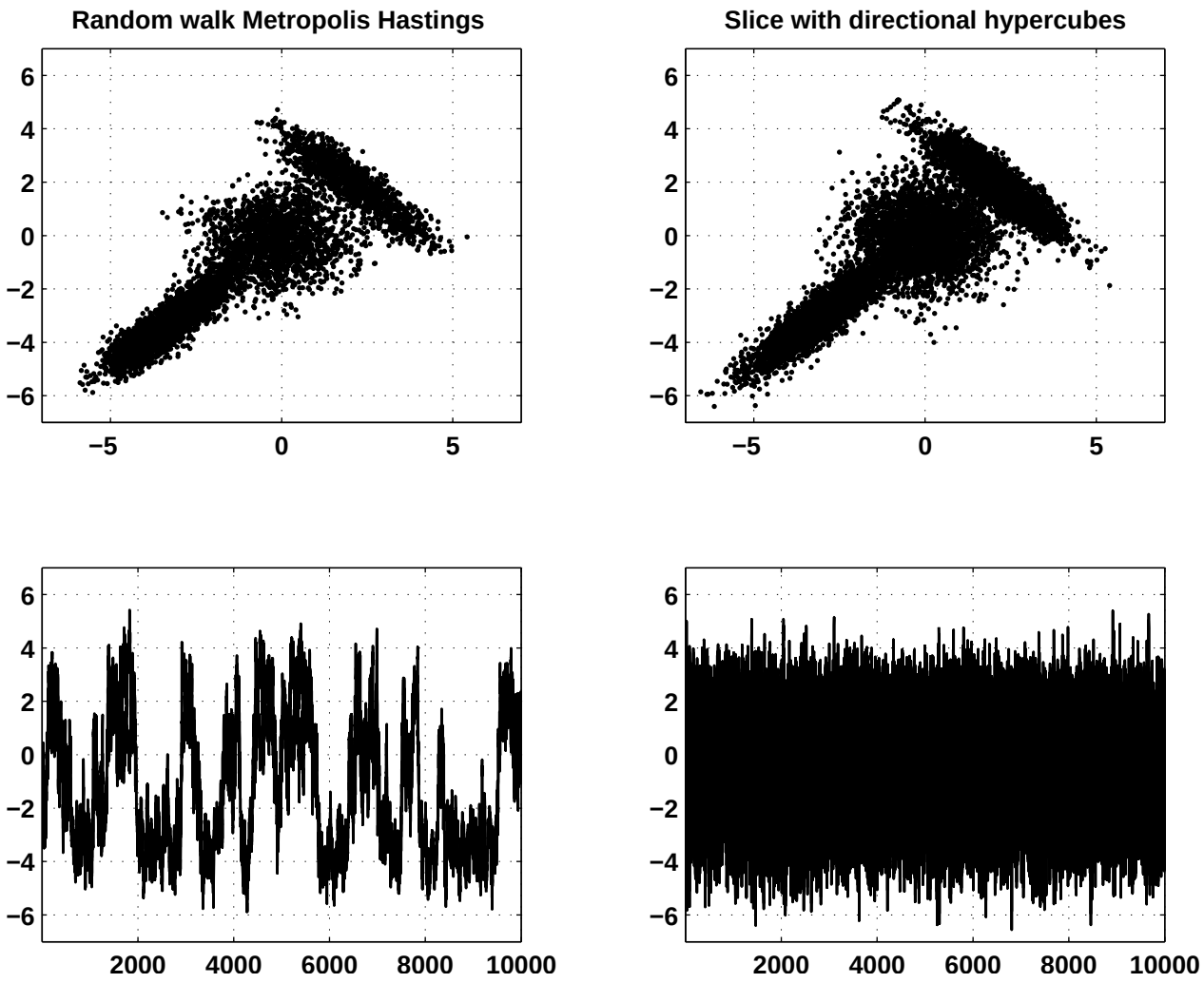

Table 1 Sampler efficiency - Marron and Wand densities

	RW-MH	Slice											
		Stepping out				Doubling				Random positioning			
W		$\frac{1}{2}\sigma$	3σ	10σ	100σ	$\frac{1}{2}\sigma$	3σ	10σ	100σ	$\frac{1}{2}\sigma$	3σ	10σ	100σ
						Skewed							
IF	5.13	1.2	1.21	1.2	1.17	1.26	1.21	1.23	1.23	82.94	5.7	1.84	1.24
Eval	2	10.18	5.92	6.34	9.6	25.29	15.74	9.77	9.81	2.11	2.66	3.77	7.48
RE	1	1.19	0.7	0.74	1.1	3.1	1.86	1.17	1.17	17.1	1.48	0.68	0.91
						Strongly skewed							
IF	10.7	3.11	3.08	3.09	3.07	3.2	3.2	3.18	3.17	100.32	8.6	3.94	3.16
Eval	2	8.46	6.29	7.23	10.87	21.87	12.99	9.52	10.92	2.34	3.38	4.81	8.78
RE	1	1.23	0.91	1.04	1.56	3.27	1.94	1.41	1.62	10.96	1.36	0.89	1.3
						Kurtotic							
IF	4.86	0.97	0.97	0.97	0.97	1.05	0.99	1.01	0.98	93.32	6.26	1.68	1.04
Eval	2	9.92	6.41	7.07	10.49	22.56	14.67	10.16	10.69	2.31	3.27	4.57	8.39
RE	1	0.99	0.64	0.71	1.05	2.45	1.5	1.05	1.08	22.22	2.11	0.79	0.9
						Outlier							
IF	8.15	0.97	0.98	0.98	0.96	1.09	1.05	1.03	1	100.24	23.4	3.78	1.09
Eval	2	8.16	6.35	7.44	11.2	21.39	11.5	9.5	11.39	2.3	3.47	5.07	9.13
RE	1	0.48	0.38	0.45	0.66	1.42	0.74	0.6	0.7	14.12	4.98	1.17	0.61
						Bimodal							
IF	4.93	1.26	1.12	1.07	1.05	1.24	1.15	1.09	1.06	84.18	4.04	1.5	1.1
Eval	2	10.33	5.92	6.2	9.35	25.72	16.35	9.89	9.44	2.12	2.6	3.59	7.23
RE	1	1.32	0.67	0.67	1	3.23	1.9	1.1	1.02	18.11	1.06	0.55	0.81
						Separate Bimodal							
IF	9.18	22.13	2.92	2.17	1.96	5.82	3.85	2.14	1.94	148.43	6.31	2.55	1.95
Eval	2	6.86	6.19	6.93	10.46	22.64	13.64	9.28	10.51	2.31	3.26	4.5	8.37
RE	1	8.28	0.98	0.82	1.12	7.18	2.86	1.08	1.11	18.67	1.12	0.62	0.89

Notes: σ is the standard deviation implied by the density under study, $Eval$ is the average number of evaluations, $IF = 1 + 2\sum_j \rho_j$ is the inefficiency factor, $RE = (IF_A \times Eval_A)/(IF_{MH} \times Eval_{MH})$ is the relative efficiency of sampler A with respect to the RW-MH sampler

Table 1 Sampler efficiency - Marron and Wand densities, cont'd

	RW-MH	Slice											
		Stepping out				Doubling				Random positioning			
W		$\frac{1}{2}\sigma$	3σ	10σ	100σ	$\frac{1}{2}\sigma$	3σ	10σ	100σ	$\frac{1}{2}\sigma$	3σ	10σ	100σ
						Skewed Bimodal							
IF	5.11	1.22	1.18	1.2	1.19	1.27	1.25	1.23	1.23	82.86	4.52	1.68	1.22
Eval	2	10.36	5.92	6.25	9.44	25.53	16.07	9.84	9.53	2.12	2.62	3.65	7.32
RE	1	1.24	0.68	0.73	1.1	3.18	1.97	1.18	1.14	17.18	1.16	0.6	0.88
						Trimodal							
IF	5.24	1.41	1.21	1.13	1.11	1.38	1.24	1.16	1.14	87.98	4.11	1.58	1.16
Eval	2	10.14	5.94	6.24	9.42	25.41	16.23	9.86	9.51	2.15	2.64	3.65	7.3
RE	1	1.37	0.69	0.67	1	3.34	1.92	1.09	1.03	18.04	1.04	0.55	0.81
						Claw							
IF	5.07	1.53	1.2	1.14	1.13	1.6	1.24	1.19	1.13	90.13	4.97	1.71	1.15
Eval	2	9.81	6.1	6.54	9.82	23.31	15.71	9.82	9.91	2.35	2.9	3.99	7.7
RE	1	1.48	0.72	0.74	1.09	3.67	1.92	1.15	1.11	20.9	1.42	0.67	0.88
						Double Claw							
IF	4.99	1.27	1.12	1.08	1.06	1.26	1.16	1.1	1.08	85.55	4.11	1.55	1.11
Eval	2	10.33	5.97	6.26	9.43	25.62	16.32	9.93	9.52	2.18	2.66	3.66	7.3
RE	1	1.32	0.67	0.68	1	3.22	1.9	1.1	1.03	18.7	1.1	0.57	0.82
						Asymmetric claw							
IF	5.61	3.09	1.38	1.24	1.2	2.78	1.44	1.25	1.22	89.83	4.82	1.74	1.24
Eval	2	9.72	6.05	6.44	9.69	24.34	15.54	9.83	9.78	2.29	2.84	3.88	7.57
RE	1	2.68	0.74	0.71	1.03	6.04	1.99	1.1	1.06	18.31	1.22	0.6	0.84
						Smooth comb							
IF	11.99	15.84	2.97	2.19	1.97	6.8	3.58	2.23	1.97	144.07	6.23	2.58	2.02
Eval	2	6.96	6.35	7.09	10.64	20.78	13.44	9.36	10.68	2.59	3.46	4.68	8.56
RE	1	4.59	0.78	0.65	0.87	5.89	2.01	0.87	0.88	15.56	0.9	0.5	0.72

Notes: σ is the standard deviation implied by the density under study, *Eval* is the average number of evaluations, $IF = 1 + 2\sum_j \rho_j$ is the inefficiency factor, $RE = (IF_A \times Eval_A)/(IF_{MH} \times Eval_{MH})$ is the relative efficiency of sampler A with respect to the RW-MH sampler

Table 2 Sampler efficiency - uncorrelated variables

	RW-MH	Slice			
		One-at-a-time	Plain	Gradient	Directional
		d=2			
Max IF	9.8	0.96	1.55	1.14	1.81
Eval	2	11.81	10.98	13.77	11.71
RE	1	0.58	0.87	0.8	1.08
		d=5			
Max IF	21.63	0.97	4.93	2.15	8.48
Eval	2	29.54	63.34	89.52	65.3
RE	1	0.66	7.22	4.45	12.79
		d=10			
Max IF	41.32	0.98	13.31	6.71	46.06
Eval	2	59.08	1881	2127	1869
RE	1	0.7	302.9	172.8	1042

Notes: $Eval$ is the average number of evaluations, $IF = 1 + 2\sum_j \rho_j$ is the inefficiency factor, $RE = (IF_A \times Eval_A)/(IF_{MH} \times Eval_{MH})$ is the relative efficiency of sampler A with respect to the RW-MH sampler, d is the dimension of θ.

Table 3 Sampler efficiency - highly correlated variables

	RW-MH	Slice			
		One-at-a-time	Plain	Gradient	Directional
		d=2			
Max IF	346.0	19.07	28.67	19.47	1.08
Eval	2	12.12	10.47	14.8	11.71
RE	1	0.33	0.43	0.42	0.02
		d=5			
Max IF	699.4	67.35	226.1	144.1	1.56
Eval	2	30.31	47.5	71.3	65.31
RE	1	1.46	7.68	7.35	0.07
		d=10			
Max IF	698.8	127.9	304.8	279.3	3.03
Eval	2	60.29	1254	1379	1884
RE	1	5.52	273.6	275.7	4.09

Notes: $Eval$ is the average number of evaluations, $IF = 1 + 2\sum_j \rho_j$ is the inefficiency factor, $RE = (IF_A \times Eval_A)/(IF_{MH} \times Eval_{MH})$ is the relative efficiency of sampler A with respect to the RW-MH sampler, d is the dimension of θ.

Tab 4 Sampler efficiency - bivariate mixture of Normals

	RW-MH	Slice			
		One-at-a-time	Plain	Gradient	Directional
Max IF	231.31	22.2	23.31	14.27	3.02
Eval	2	12.67	10.39	15.93	13.71
RE	1	0.61	0.52	0.49	0.09

Notes: $Eval$ is the average number of evaluations, $IF = 1 + 2\sum_j \rho_j$ is the inefficiency factor, $RE = (IF_A \times Eval_A)/(IF_{MH} \times Eval_{MH})$ is the relative efficiency of sampler A with respect to the RW-MH sampler.

Univariate slice sampler: the MatLab code

The function SLICESTEPOUT.M implements the univariate version of the slice sampler with stepping out described so far. It returns a simulated draw (XSIM) and the number of calls NEVAL to the function FUNC. FUNC should be provided by the user and takes two input: THETA, and LAMBDA. The latter is a vector of extra hyperparameters need to evaluate FUNC. FUNC returns the value of the function evaluated at THETA. XLB and XUB represent the lower and upper bounds of the variable THETA. Set it to plus or minus **Inf** in case the variable has an unlimited support. Finally % indicates comments.

```
function [xsim, neval] = slicestepout(xold, W, XLB, XUB, lambda)
neval = 0;

% 1. Draw z = ln[f(x0)] - exp(1) where exp(1)=-ln(U(0,1))
%    This defines the slice S = {x: z < ln(f(x))}
fxold = func(xold, lambda);
fxold = log(fxold);
neval = neval + 1;
z = fxold + log(rand(1,1));

% 2. Find I = (L,R) around x0 that contains S as much as possible
%    using the stepping out procedure.
u = rand(1,1);
L = max(XLB, xold - W*u);
R = min(XUB, L + W);
while(L > XLB)
    xsim = L;
    fxl = func(xsim, lambda);
    fxl = log(fxl);
    neval = neval + 1;
    if (fxl <= Z)
        break;
    end
    L = max(XLB, L - W);
end
while(R < XUB)
    xsim = R;
```

```
    fxr = func(xsim, lambda);
    fxr = log(fxr);
    neval = neval + 1;
    if (fxr <= Z)
        break;
    end
    R = min(XUB, R + W);
end

% 3. Sampling from the set A = (I intersect S)
fxsim = Z-1;
while (fxsim < Z)
    u = rand(1,1);
    xsim = L + u*(R - L);
    fxsim = func(xsim, lambda);
    fxsim = log(fxsim);
    neval = neval + 1;
    if (xsim > xold)
        R = xsim;
    else
        L = xsim;
    end
end
```