THE FUSS ALGORITHM: A FAST UNIVERSAL SELF-TUNED SAMPLER WITHIN GIBBS

L. Martino[1], H. Yang[2], D. Luengo[3], J. Kanniainen[2], J. Corander[1]

[1] Dep. of Mathematics and Statistics, University of Helsinki, 00014 Helsinki (Finland).
[2] Dep. of Industrial Engineering, Tampere University of Technology, Tampere (Finland).
[3] Dep. of Circuits and Systems Engineering, Universidad Politécnica de Madrid, 28031 Madrid (Spain).

ABSTRACT

Gibbs sampling is a well-known Markov Chain Monte Carlo (MCMC) technique, widely applied to draw samples from multivariate target distributions which appear often in many different fields (machine learning, finance, signal processing, etc.). The application of the Gibbs sampler requires being able to draw efficiently from the univariate full-conditional distributions. In this work, we present a simple, self-tuned and extremely efficient MCMC algorithm that produces virtually independent samples from the target. The proposal density used is self-tuned to the specific target but it is not adaptive. Instead, the proposal is adjusted during the initialization stage following a simple procedure. As a consequence, there is no "fuss" about convergence or tuning, and the execution of the algorithm is remarkably speed up. Although it can be used as a stand-alone algorithm to sample from a generic univariate distribution, the proposed approach is particularly suited for its use within a Gibbs sampler, especially when sampling from spiky multi-modal distributions. Hence, we call it FUSS (Fast Universal Self-tuned Sampler). Numerical experiments on several synthetic and real data sets show its good performance in terms of speed and estimation accuracy.

Keyword: Markov Chain Monte Carlo; Gibbs sampling; adaptive Metropolis rejection sampling; Bayesian inference.

1. INTRODUCTION

Bayesian methods and their implementations by means of sophisticated Monte Carlo techniques [14, 23] have become very popular over the last years. Indeed, many practical problems demand procedures for sampling from probability distributions with non-standard forms, such as Markov chain Monte Carlo (MCMC) methods [4] and particle filters [2]. MCMC techniques generate samples from a target probability density function (pdf) by drawing from a simpler proposal pdf [13, 14]. The two best known MCMC approaches are the Metropolis-Hastings (MH) algorithm and the Gibbs sampler [23]. The Gibbs sampling technique is extensively used in Bayesian inference [23], e.g., in signal processing [24] and machine learning [20], to generate samples from multi-dimensional target densities, drawing each component of the generated samples from the corresponding univariate full-conditional density.

The key point for its successful application is being able to draw efficiently from these univariate pdfs. The best scenario for Gibbs sampling occurs when exact samplers for each full-conditional are available. Otherwise, sampling techniques like rejection sampling (RS) or MH-type algorithms are used *within* the Gibbs sampler to draw from complicated full-conditionals. In the first case, samples generated from the RS algorithm are independent, but the acceptance rate can be very low. In the second case, we have an approach where an MCMC, the MH method, is applied *inside* another MCMC, that is the Gibbs samplers. Therefore, the typical problems of the *external*-MCMC (long "burn-in" period, large correlation, etc.) could raise dramatically if the *internal*-MCMC is not extremely efficient. Although the Gibbs sampler needs only one sample from each full-conditional, in this case several iterations are necessary to avoid the "burn-in" period of the *internal*-MCMC. The length of the "burn-in" period is strictly related to the correlation among the samples. Higher correlation corresponds to a slower convergence of the chain.

Thus, several automatic and self-tuning samplers, such as *adaptive rejection sampling* (ARS) [5, 8], *Adaptive Rejection Metropolis Sampling* (ARMS) [6, 7, 19], *Independent Doubly Adaptive Rejection Metropolis Sampling* (IA2RMS) [18], and *Adaptive Sticky Metropolis* [16] have been proposed. All of these methods build an adaptive sequence of proposal pdfs via some interpolation procedure given a set of support points. The proposal is updated when a new support point is incorporated, according to some statistical criterion. Although their performance can be extremely good, the results show a dependence from the initial set of support points. Moreover, depending on the complexity of the target, these algorithms can appear too

slow. Another drawback is the need of take care of the ergodicity especially in the applications within Gibbs sampling [7, 23]. Other related works, where a non-adaptive proposal pdf is built via interpolation procedures, can be found in literature [21, 25]. Furthermore, owing their good performance, different kind of adaptive MH methods based on an independent proposal have been also studied [10, 11].

In this work, we present a novel algorithm that uses a different strategy: start with a huge number of support points and then remove some of them, according to certain conditions. The resulting method is extremely fast (it fits particularly in a MATLAB implementation) and extremely efficient (it yields virtually independent samples), as shown in the numerical results, even with highly multimodal and complicated targets. The dependence on the initial set of points is drastically reduced, since the user must provide only a large interval where he considers that the algorithm should concentrate the main computational effort. Moreover, the proposal is self-tuned, during the initialization stage, but non-adaptive afterwards. Hence, ergodicity is not an issue and the convergence of the chain to the target distribution is always guaranteed. For these reasons, we call the new method as *FUSS algorithm* ("Fast Universal Self-tuned Sampler") since, with this sampler, there is no "fuss" about convergence or tuning. The FUSS algorithm is particularly advisable for multimodal spiky target densities, i.e., densities with several sharp and tight modes, where virtually all of the existent MCMC techniques often fail. This kind of target pdfs often appears in several applications, as in financial inference problems (see Section 6 for an example).

The rest of the paper is divided as follows. Sections 2-3 are devoted to recall the general framework and describe the structure of the novel technique. The details about the proposal construction and generation are given in Section 4. Then, different pruning algorithms are introduced in Section 5. Section 6 provides different numerical simulations. Finally, Section 7 contains some brief final considerations.

2. PROBLEM STATEMENT

Bayesian inference often requires drawing samples from complicated multivariate posterior pdfs, $\pi(\mathbf{x}|\mathbf{y})$ with $\mathbf{x} \in \mathcal{X}^D \subseteq \mathbb{R}^D$. A common approach, when direct sampling from $\pi(\mathbf{x}|\mathbf{y})$ is unfeasible, is using a Gibbs sampler [23]. At the i-th iteration, a Gibbs sampler obtains the d-th component ($d = 1, \ldots, D$) of $\mathbf{x}$, x_d, drawing from the full conditional pdfs of x_d given all the previous generated components [23, 3, 12], i.e.,

$$x_d^{(i)} \sim \bar{\pi}(x_d|\mathbf{x}_{1:d-1}^{(i)}, \mathbf{x}_{d:D}^{(i-1)}) = \bar{\pi}(x_d) \propto \pi(x_d), \quad x_d \in \mathcal{X}, \tag{1}$$

with the initial vector drawn from the prior, i.e., $\mathbf{x}^{(0)} \sim \bar{\pi}_0(\mathbf{x})$. However, even sampling from the univariate pdf in Eq. (1) can often be complicated. In these cases, a common approach is using another Monte Carlo technique (e.g., rejection sampling (RS) or the Metropolis-Hastings (MH) algorithms) within the Gibbs sampler, drawing candidates from a simpler proposal,

$$\bar{p}(x) \propto p(x) = e^{W(x)}, \quad x \in \mathbb{R}.$$

The best case is when an RS technique can be applied since it yields independent and identically distributed (i.i.d.) samples. However, the RS technique requires that $p(x) \geq \pi(x)$ for all $x \in \mathcal{X}$. In general, it is not straightforward to satisfies this inequality for: instance, the adaptive rejection sampling (ARS) technique can be applied only for log-concave target pdfs. Thus, in general, the use of another MCMC method becomes mandatory. In this case, the performance of this approach depends strictly on the choice of $\bar{p}(x)$. For sake of simplicity, in the sequel we denote the univariate target pdf (i.e., the full-conditional proposal in Eq. (1)) as $\bar{\pi}(x)$.

Our aim is designing an efficient fast sampler to draw from the univariate target pdf,

$$\bar{\pi}(x) \propto \pi(x) = e^{V(x)}, \quad x \in \mathcal{X} \subseteq \mathbb{R}, \tag{2}$$

where $\pi(x)$ is unnormalized and $V(x) = \log[\pi(x)]$.

3. STRUCTURE OF THE ALGORITHM

The FUSS algorithm is an MCMC based on an independent proposal pdf built via a simple interpolation procedure, shown in the next section. The general structure is given in Table 1. The first 3 steps form a pre-processing procedure, applied only once, useful to obtain a good proposal density. The step 4 contains the MCMC iterations, repeated K times. Here, wee consider two possible techniques for the step 4 of FUSS:

- *The Metropolis-Hastings (MH) algorithm [23]:* it is shown in Table 2. In this case, we denote the whole method as *FUSS-MH*.

Table 1. **General structure of the FUSS algorithm.**

1. **Initialization:** Choose a set of support points $\mathcal{S}_M = \{s_1, \ldots, s_M\}$, sorted in ascending order $s_1 < s_2 < \ldots < s_M$, and the total number of desired sample K.

2. **Pruning:** Remove certain support points according to some criterion, providing a new set $\mathcal{S}_m$ with $m < M$.

3. **Construction:** Build adequately a proposal function $p(x|\mathcal{S}_m)$ given $\mathcal{S}_m$, using a suitable procedure.

4. **MCMC algorithm:** apply K steps of an MCMC method using $p(x|\mathcal{S}_m)$ as proposal pdf and yielding a set of samples $\{x_1, \ldots, x_K\}$.

Table 2. **Possible step 4 of FUSS: the Metropolis-Hastings method**

3.1 Set $k = 0$ and choose x_0.

3.2 Draw $x' \sim \bar{p}(x) \propto p(x|\mathcal{S}_m)$ and $u' \sim \mathcal{U}([0, 1])$.

3.4 Set $x_{k+1} = x'$ with probability

$$\alpha_{MH} = 1 \wedge \frac{\pi(x')p(x_k|\mathcal{S}_m)}{\pi(x_k)p(x'|\mathcal{S}_m)}, \tag{3}$$

otherwise, with probability $1 - \alpha_{MH}$ set $x_{k+1} = x_k$.

3.5 If $k \leq K$, set $k = k + 1$ and repeat from step 3.2. Otherwise, stop.

Table 3. **Possible step 4 of FUSS: the rejection chain method**

3.1 Set $k = 0$ and choose x_0.

3.2 Draw $x' \sim \bar{p}(x) \propto p(x|\mathcal{S}_m)$ and $u' \sim \mathcal{U}([0, 1])$.

3.3 If $u' \geq \frac{\pi(x')}{p(x'|\mathcal{S}_m)}$ repeat from step 3.2.

3.4 If $u' \leq \frac{\pi(x')}{p(x'|\mathcal{S}_m)}$, with probability

$$\alpha_{RC} = 1 \wedge \frac{\pi(x')\left[\pi(x_k) \wedge p(x_k|\mathcal{S}_m)\right]}{\pi(x_k)\left[\pi(x') \wedge p(x'|\mathcal{S}_m)\right]}, \tag{4}$$

set $x_{k+1} = x'$, otherwise, with probability $1 - \alpha_{RC}$ set $x_{k+1} = x_k$.

3.5 If $k \leq K$, set $k = k + 1$ and repeat from step 3.2. Otherwise, stop.

- *The Rejection chain algorithm [26, 27]:* it is shown in Table 3. In this case, firstly a rejection sampling (RS) test is performed; whether a sample is accepted then a MH step is applied to ensure to drawing from the target pdf. We denote the complete method as *FUSS-RC*.

FUSS-RC is slower than FUSS-MH since when a sample is rejected in the RS test the chain is not moved forward. On the other hand, FUSS-RC yields samples with less correlation due to application of the RS test. We will test and compare the performance in the numerical simulations. The notation $a \wedge b$ denotes the minimum between two real values, i.e., $\min[a, b]$.

3.1. Important remarks

It is important to stand out the following considerations:

- If $p(x|\mathcal{S}_m) \geq \pi(x)$ for all $x \in \mathcal{X}$ then the FUSS-RC algorithm becomes a rejection sampler, providing i.i.d. samples from $\bar{\pi}(x)$. Indeed, note that in this scenario the probability α_{RC} of accepting the new state is always 1, i.e., after passing the rejection test, the sample is automatically accepted.

- Observe that in FUSS-RC, after passing the rejection test, the samples are distributed as

$$\bar{q}(x) \propto q(x) = \pi(x) \wedge p(x|\mathcal{S}_m).$$

  Thus, $q(x)$ is used as proposal in the acceptance function α_{RC}. Note also that $q(x)$ is closer to $\pi(x)$ than $p(x|\mathcal{S}_m)$: this is the reason why FUSS-RC produces samples with less correlation than FUSS-MH.

- All the operations in both algorithms, FUSS-MH and FUSS-RC, can be easily implemented in log-domain evaluating only the functions $V(x) = \log[\pi(x)]$ and $W(x) = \log[p(x|\mathcal{S}_m)]$. In FUSS-MH, the acceptance probability can be expressed as

$$\alpha_{MH} = \exp\Big(V(x') + W(x_k) - V(x_k) - W(x')\Big).$$

  Whereas in FUSS-RC, the ratio in the RS test can be written $e^{W(x') - V(x')}$ and the probability of accepting a new state can be expressed as

$$\alpha_{RC} = \exp\Big(V(x') + V(x_k) \wedge W(x_k) - V(x_k) - V(x') \wedge W(x')\Big).$$

- Steps from 1 to 3 of the general FUSS algorithm in Table 1 are performed only once. The success of the FUSS algorithms lies on the speed in performing these steps and the quality of the final proposal density. Indeed, if the final proposal has a shape close to $\pi(x)$ the generated samples will be virtually independent.

- The initial support points in $\mathcal{S}_M$ plays the role of parameters of the FUSS algorithms. After the pruning step, the proposal $p(x|\mathcal{S}_m)$ is built according to the new set $\mathcal{S}_m$ with $m < M$. After that, the proposal p is kept fixed. Thus, the FUSS techniques are standard non-adaptive MCMC, avoiding any issue about the ergodicity.

- The possibility of applying FUSS directly for drawing from multidimensional distributions depends on the ability to construct efficiently the proposal pdf via interpolation in dimensions higher than one (step 3 in the general structure of FUSS). In this work, we consider the application of FUSS within Gibbs sampling, drawing from univariate full-conditional pdfs.

3.2. Initialization and general strategy for FUSS

The initial set $\mathcal{S}_M$ should be cover the regions of high probabilities described by the target $\pi(x)$. In general, if no prior information is available we suggest the following FUSS approach:

- Choose a huge, thin, initial (uniform) grid of support points, $s_{i+1} - s_i = \epsilon$, i.e.,

$$\mathcal{S}_M = \{s_1, s_2 = s_1 + \epsilon, \ldots, s_M\},$$

  in order to capture all the main features of the target.

Then we reduce the number of support points according to a certain criterion obtaining a new set $\mathcal{S}_m$ (see Section 5 for some examples) and we build a stepwise approximation of the target pdf given the pruned support points (see Section 4). The resulting proposal pdf is a self-tuned but non-adaptive, since it does not vary during the run of the chain (it is "adapted" offline). In the next section, we describe in details the construction of the proposal.

4. CONSTRUCTION OF THE PROPOSAL DENSITY

In several applications, it is useful to evaluate the target pdf in the log-domain so that, here, we consider the construction of the proposal function in the log-domain as well. Let us consider a set of support points (after the pruning step),

$$\mathcal{S}_m = \{s_1, s_2, \ldots, s_m\} \subset \mathcal{X},$$

where $s_1 < \ldots < s_m$, and the intervals $\mathcal{I}_0 = (-\infty, s_1]$, $\mathcal{I}_j = (s_j, s_{j+1}]$ for $j = 1, \ldots, m_t - 1$ and $\mathcal{I}_m = (s_m, +\infty)$. Then, let us consider

$$\bar{p}(x) \propto p(x|\mathcal{S}_m) = e^{W(x)},$$

where $W(x)$ is built as in Eq. (5) using a piecewise constant approximation, with the exception of the first and last intervals corresponding to the tails. Mathematically,

$$W(x) = w_i(x) = \max\left[V(s_i), V(s_{i+1})\right] \mathbb{I}_{\mathcal{I}_i}(x), \ 1 \leq i \leq m_t - 1, \tag{5}$$

where

$$\mathbb{I}_{\mathcal{I}_i}(x) = \begin{cases} 1, & x \in \mathcal{I}_i = (s_i, s_{i+1}], \\ 0, & x \neq \mathcal{I}_i = (s_i, s_{i+1}]. \end{cases} \tag{6}$$

In the first and last interval, $\mathcal{I}_0$ and $\mathcal{I}_m$, we have

$$W(x) = w_j(x), \quad j = \{0, m\}, \quad x \in \mathcal{I}_j,$$

where $w_j(x)$ represents a generic log-tail function. For instance, choosing light tails, $w_j(x)$, $j = \{0, m\}$, are linear functions.Figures 1 and 4 provide some specific examples. For further details, see Appendix A. The choice of taking the maximum is in order to satisfy the inequality $W(x) \geq V(x)$, in more regions as possible. If this inequality is verified for all $x \in \mathcal{X}$ then also $p(x|\mathcal{S}_m) \geq \pi(x)$, and FUSS-RC becomes a standard rejection sampler providing independent samples.

(a)

(b)

(c)

(d)
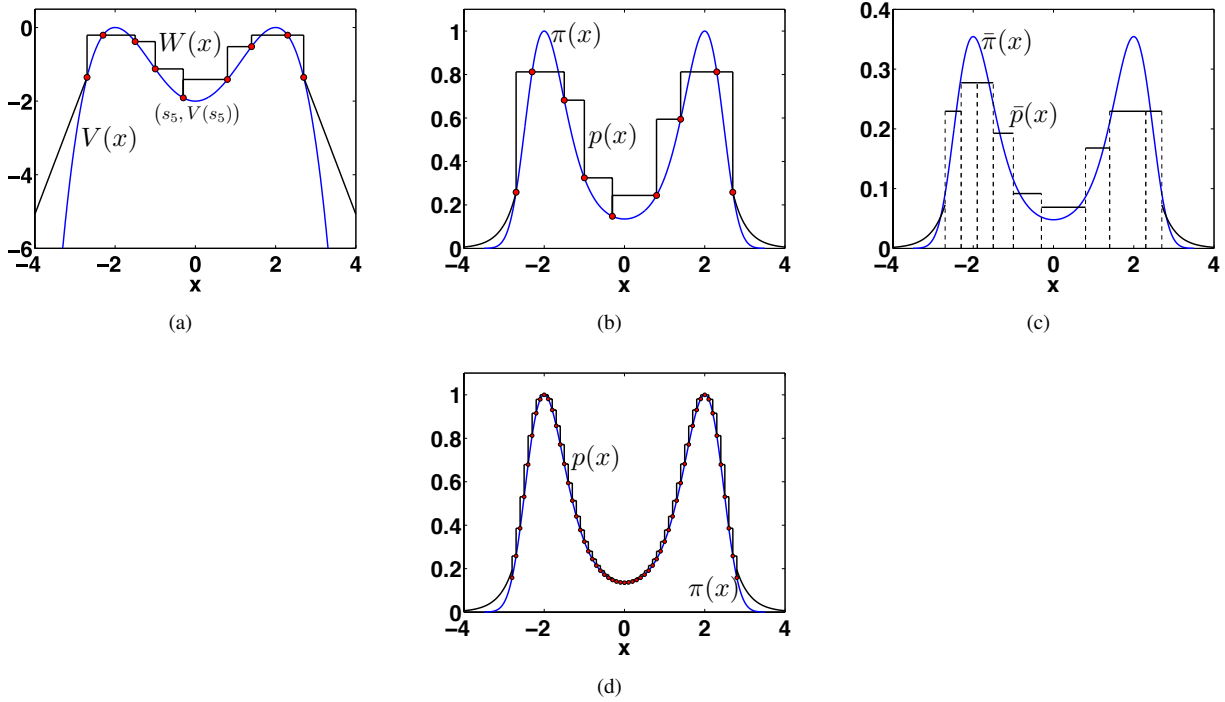
Fig. 1. Example of the proposal construction. **(a)** Construction procedure with $m = 9$ support points, in the log-domain. The log-tails are in this case light tails (two straight lines). **(b)** The corresponding unnormalized densities $p(x) = e^{W(x)}$ and $\pi(x) = e^{V(x)}$. **(c)** The corresponding normalized densities $\bar{p}(x) \propto p(x)$ and $\bar{\pi}(x) \propto \pi(x)$. **(d)** Construction of $p(x)$ and $\pi(x)$ with a thin grid of support points.

4.1. Variate generation from $\bar{p}(x)$

The proposal $\bar{p}(x) \propto p(x|\mathcal{S}_m)$ is composed, in general, by $m + 1$ pieces, including the two tails. Note that $\bar{p}(x)$ can be seen as a finite mixture

$$\bar{p}(x) \propto p(x|\mathcal{S}_m) = \sum_{i=0}^{m} \eta_i \phi_i(x),$$

<div style="text-align: center">

Table 4. **Pruning algorithm P1**

</div>

1. Given $\mathcal{S}_M = \{s_1, \ldots, s_M\}$, decide the desired number m of final support points (or a rate of reduction $\frac{m}{M}$).

2. Sort $\pi(s_i)$, $i = 1, \ldots, M$, in a decreasing order

$$\pi(s_{r_1}) = \max_{1 \le i \ge N} \pi(s_i) \ge \pi(s_{r_2}) \ge \ldots \ge \pi(s_{r_M}) = \min_{1 \le i \ge N} \pi(s_i).$$

3. Return $\mathcal{S}_m = \{s_{r_1}, s_{r_2}, \ldots, s_{r_m}\}$.

<div style="text-align: center">

Table 5. **Pruning algorithm P2**

</div>

1. Given $\mathcal{S}_M = \{s_1, \ldots, s_M\}$, choose a value $\delta \in (0, 1)$.

2. Find all the support points $s_{k_j} \in \mathcal{S}_M$ such that

$$\pi(s_{k_j}) \le \delta \max_{1 \le i \le M} \pi(s_i). \tag{7}$$

3. Return $S_m = S_M \setminus \{s_{k_1}, \ldots, s_{k_G}\}$, where G is the number of points satisfying the inequality (7).

where $\sum_{i=0}^{m} \eta_i = 1$ and

$$\phi_i(x) \propto \exp(w_i(x)), \quad x \in \mathcal{I}_i.$$

Therefore, in order to draw adequately from $\bar{p}(x)$, it is necessary:

1. Compute the area A_i below each pieces, $i = 0, \ldots, m$. It can be done analytically easily for the rectangular pieces, and for the tails considered here (exponential or Pareto, for instance) as well. Then, normalize them

$$\eta_i = \frac{A_i}{\sum_{j=1}^{m} A_j}, \quad \text{for} \quad i = 0, \ldots, m.$$

2. Choose a piece, i.e., an index $j^* \in \{0, \ldots, m\}$ according to the weights η_i, $i = 0, \ldots, m$.

3. Given the index j^*, draw $x' \sim \phi_{j^*}(x) \propto \exp(w_{j^*}(x))$.

It is important to remark that the process of calculate the areas A_i and then the weights η_i is done once, before running the Markov chain. The calculation of the areas A_i of the rectangular regions is straightforward and fast.

5. PRUNING ALGORITHMS

The computational cost of drawing from $\bar{p}(x)$ and, as a consequence, the speed of the algorithm depend on the number of support points. The application of the pruning step has two advantages: it speeds up the algorithm, and allows the use of a greater number of initial support points (capturing all the features of the target). In the sequel, we present some possible pruning criteria, sorted for increasing level of complexity. For the sake of simplicity, we assume a bounded target $\pi(x)$. The first two procedures P1 and P2 are shown in Tables 4 and 5. They are based on the simple idea of pruning all the points s_r with "small" value $\pi(s_r)$. They are the simplest and fastest ones but they also present several limitations, then they should be used carefully. For instance, they are not advisable for heavy tailed distributions. Another drawback of the procedures P1 and P2 is that the performance are quite sensitive to the dispersion of the target.

More refined pruning techniques can be easily provided. An example is the procedure P3 in Table 6. The underlying idea is that we can remove support points where the target is "almost" flat, i.e., $|\pi(s_{i+1}) - \pi(s_i)| \approx 0$. Moreover, note that at the first iteration, if a uniform grid is used, i.e., $s_{i+1} - s_i = \epsilon$, the ratio $\frac{\pi(s_{i+1}) - \pi(s_i)}{\epsilon}$ is an estimation of the first derivative, hence a condition over $|\pi(s_{i+1}) - \pi(s_i)|$ can be consider a condition over the first derivative of π. Clearly, this procedure could be repeated until achieving the desired rate of reduction or simply iterated N times. In Table 6, the procedure is iterated until the pruning condition is no longer verified.

Table 6. **Pruning algorithm P3**

1. Given $\mathcal{S}_M = \{s_1, \ldots, s_M\}$, choose a value $\delta \in (0,1)$. Set $\mathcal{S}^{(0)} = \mathcal{S}_M$, $r = 1$ and

$$L = \max_{1 \leq i \leq M} |\pi(s_{i+1}) - \pi(s_i)|.$$

2. While $G \neq 0$:

   (a) Find all the support points $s_{k_j} \in \mathcal{S}^{(n)}$ such that

$$\left|\pi(s_{k_j+1}) - \pi(s_{k_j})\right| \leq \delta L, \tag{8}$$

   (b) $\mathcal{S}^{(r)} = \mathcal{S}^{(r-1)} \setminus \{s_{k_1}, \ldots, s_{k_G}\}$ where $G = |\{s_{k_1}, \ldots, s_{k_G}\}|$.

   (c) Set $r = r + 1$.

3. Return $\mathcal{S}_m = \mathcal{S}^{(n)} \setminus \{s_1\}$.

5.1. Optimal pruning strategy

The performance of a rejection sampler or an independent Metropolis algorithm is related to the L_1 distance between the target and the proposal [23],

$$D_{p|\pi}(\mathbb{R}) = \int_{-\infty}^{\infty} |p(x) - \pi(x)| dx. \tag{9}$$

With the proposal procedure considered here, we can write

$$D_{p|\pi}(\mathbb{R}) = \sum_{j=0}^{m} D_{p|\pi}(\mathcal{I}_j),$$

where $D_{p|\pi}(\mathcal{I}_j)$ denotes the local L_1 distance within the i-th interval ($0 \leq j \leq m$), i.e.

$$D_{p|\pi}(\mathcal{I}_j) = \int_{\mathcal{I}_j} |p(x) - \pi(x)| dx = \int_{s_j}^{s_{j+1}} |p(x) - \pi(x)| dx. \tag{10}$$

Here, for the sake of simplicity, we are considering the (non-rigorous) definition $s_0 = -\infty$ and $s_{m+1} = +\infty$, since $\mathcal{I}_0 = (-\infty, s_1]$ and $\mathcal{I}_m = [s_m, \infty)$. Moreover, recall that $s_1 < s_2 < \ldots < s_m$.

The essential consideration is the following: when a support point is removed, the distance between the target and the proposal generally will tend to increase, thus leading to a worse performance of the algorithm. Hence, an optimal criterion for pruning support points is discarding those that lead to an increase, as small as possible, in the L_1 distance between $p(x)$ and $\pi(x)$. Since, in this work, the proposal $p(x)$ is a piecewise constant function with values $\exp(w_i) = \exp(\max[V(s_i), V(s_{i+1})])$ (with the exception of the tails), and considering a *continuous* target pdf, it is apparent that

$$D_{p|\pi}(\mathcal{I}_j) \leq B_{j,j+1} = (s_{j+1} - s_j)|\pi(s_{j+1}) - \pi(s_j)|.$$

i.e., $B_{j,j+1}$ is an upper bound for the L_1 distance $D_{p|\pi}(\mathcal{I}_j)$. If the number m of used support points grows, then clearly $B_{j,j+1} \to D_{p|\pi}(\mathcal{I}_j)$. Thus, the value $B_{j,j+1}$ can be considered as a rough approximation of $D_{p|\pi}(\mathcal{I}_j)$. This observation is the theoretical basis of the pruning strategy detailed in Table 7: consider a generic interval $[s_j, s_{j+2}]$. Depending on the approximation $B_{j,j+2}$ we decide if pruning s_{j+1} or not. Clearly, at each iteration at most $R = \lfloor \frac{M-1}{2} \rfloor$ points can be removed. The procedure is iterated until no more points are pruned, i.e., when all the upper bounds (b_r, in the notation of Table 7) are greater than the threshold δ. Clearly, the algorithm could be stopped before.

6. NUMERICAL SIMULATIONS

In this section, we first test the performance of the FUSS algorithm comparing with other standard MCMC methods drawing from univariate densities (Section 6.1). Then, we consider the application of the FUSS technique within the Gibbs sampling for drawing from multidimensional pdfs (Sections 6.2-**??**).

Table 7. **Pruning algorithm P4**

---

1. Choose a value $\delta \in (0,1)$. Given $\mathcal{S}_M = \{s_1, ..., s_M\}$, set $\mathcal{S}^{(0)} = \mathcal{S}_M$, $m = M$, $n = 0$ and

$$L = \max_{1 \leq j \leq \left\lfloor \frac{m-1}{2} \right\rfloor} (s_{2j+1} - s_{2j-1})|\pi(s_{2j+1}) - \pi(s_{2j-1})|.$$

2. For $r = 1, \ldots, R = \left\lfloor \frac{m-1}{2} \right\rfloor$:

   (a) Compute $b_r = (s_{2r+1} - s_{2r-1})|\pi(s_{2r+1}) - \pi(s_{2r-1})|$.

   (b) If $b_r \leq \delta L$, set $\mathcal{S}^{(r)} = \mathcal{S}^{(r-1)} \setminus \{s_{2r}\}$ and $n = n + 1$.

   (c) Otherwise, if $b_r > \delta$, set $\mathcal{S}^{(r)} = \mathcal{S}^{(r-1)}$.

3. If $n > 0$ set $n = 0$, $\mathcal{S}^{(0)} = \mathcal{S}^{(R)}$, $m = |\mathcal{S}^{(R)}|$ and repeat from step 2.

4. Otherwise, if $n = 0$, return $\mathcal{S}_m = \mathcal{S}^{(R)}$.

6.1. Univariate densities

6.1.1. Unimodal target pdf: Nakami distribution

First of all we consider a Nakagami target distribution, i.e.,

$$\bar{\pi}(x) \propto \pi(x) = x^{2\beta-1} \exp\left(-\frac{\beta}{\Omega}x^2\right), \quad x > 0, \ \beta \geq 0.5, \ \Omega > 0. \tag{11}$$

The Nakagami distribution is widely used for the simulation of fading channels in wireless communications [1, 15, 22]. When β is an integer or half-integer(i.e., $\beta = \frac{n}{2}$ with $n \in \mathbb{N}$), independent samples can be directly generated through the square root of a sum of squares of n zero-mean i.i.d. Gaussian random variables. However, for generic values of β there is not direct method to sample from it. Here, we consider the goal of estimating the expected value of $X \sim \bar{\pi}(x)$, $\mu = E[X] = \frac{\Gamma(\beta+\frac{1}{2})}{\Gamma(\beta)}\sqrt{\frac{\Omega}{\beta}}$, and the variance, $\sigma^2 = \Omega\left(1 - \frac{1}{\beta}\left(\frac{\Gamma(\beta+\frac{1}{2})}{\Gamma(\beta)}\right)^2\right)$, with $\Omega = 1$ and $\beta = 4.6$.

We performs the FUSS methods using different pruning procedures P2, P3, P4 with different values of the threshold parameter δ. We use an initial set $\mathcal{S}_M = \{0.01, 0.02, 0.03, \ldots, 10^3\}$ with $M = 10^5$ points. We also test a standard MH technique [14, 23] with a random walk proposal $\bar{p}(x_k|x_{k-1}) \propto \exp\{\frac{-(x_k-x_{k-1})^2}{2\sigma_p^2}\}$ with different values of σ_p. Finally, we consider another well-known methodology the *slice sampling* technique [23, Chapter 8]. For as fair as possible comparison of the wasted time, we have used for both the corresponding Matlab functions directly provided by MathWorks (`mhsample.m` and `slicesample.m`).

For all these techniques, we choose $x_0 \in \mathcal{U}[0, 10]$, set $K = 5000$ and consider all the generated samples without removing any burn-in period. We have performed $3 \cdot 10^4$ independent runs and the results are shown in Tables 8-9. These tables provides the Mean Square Errors (MSE) in the estimation of μ and σ^2, the linear correlation at lag-1 ($\rho(1)$) among the samples, the acceptance rate ($0 \leq$AR≤ 1) in the rejection sampling (RS) step, the number of points m after the pruning and the spent time. The time values are normalized w.r.t. the time spent by the MH method. Due to only FUSS-RC has an RS step, in the other algorithms AR is considered 1 since no samples are discarded. This means, the total number of iterations if FUSS-RC are greater $K = 5000$ (depending on the acceptance rate), whereas for the other methods are exactly $K = 5000$.

We can see that both FUSS algorithms always outperform the standard MH and slice techniques and they are also faster.

Both FUSS algorithms virtually reaches the performance of an *exact sampler* in the estimation of μ using independent samples, that is

$$\text{MSE}(\mu) \geq \text{MSE}_{ind}(\mu) = \frac{\sigma^2}{K} = 1.0560 \ 10^{-5},$$

FUSS-MH clearly is always faster than FUSS-RC since the lack of rejection sampling test. On the other hand, for the same reason FUSS-RC provides better results (with some exceptions with the pruning P2)[1]. Note that, in spite of the greater number

---

[1]FUSS-RC always has less correlations among samples than FUSS-MH and, as a consequence, less variance in the estimation, but in some cases it can have more bias.

Pruning	FUSS-MH				FUSS-RC			
	$\delta= 0.9$	$\delta= 0.5$	$\delta= 0.3$	$\delta= 0.01$	$\delta= 0.9$	$\delta= 0.5$	$\delta= 0.3$	$\delta= 0.01$
P2 MSE(μ)	1.1466	0.4067	0.0501	$1.09\ 10^{-5}$	1.1621	0.4444	0.0513	$1.05\ 10^{-5}$
MSE(σ^2)	0.8679	0.1692	0.0171	$1.05\ 10^{-6}$	0.8692	0.1835	0.0172	$1.09\ 10^{-6}$
$\rho(1)$	0.9315	0.6880	0.1720	0.0046	0.8791	0.6837	0.1406	$-3.08\ 10^{-4}$
AR	1	1	1	1	0.5211	0.8912	0.9516	0.9829
m	22	55	72	138	22	55	72	138
Time	0.6683	0.6742	0.6781	0.6875	1.2776	0.7947	0.7527	0.7328
P3 MSE(μ)	0.0026	$1.66\ 10^{-4}$	$1.21\ 10^{-4}$	$1.06\ 10^{-5}$	0.0019	$3.00\ 10^{-4}$	$8.23\ 10^{-4}$	$1.05\ 10^{-5}$
MSE(σ^2)	$6.51\ 10^{-4}$	$5.94\ 10^{-5}$	$6.11\ 10^{-5}$	$1.13\ 10^{-6}$	$3.02\ 10^{-5}$	$1.09\ 10^{-6}$	$8.56\ 10^{-5}$	$1.10\ 10^{-6}$
$\rho(1)$	0.0317	0.0141	0.0089	0.0053	0.0091	$2.95\ 10^{-4}$	$4.42\ 10^{-4}$	$-2.14\ 10^{-4}$
AR	1	1	1	1	0.9570	0.9630	0.9787	0.9830
m	50	88	106	166	50	88	106	166
Time	0.6712	0.6816	0.6859	0.7019	0.7676	0.7408	0.7424	0.7426
P4 MSE(μ)	$1.10\ 10^{-5}$	$1.09\ 10^{-5}$	$1.06\ 10^{-5}$	$1.06\ 10^{-5}$	$1.10\ 10^{-5}$	$1.09\ 10^{-5}$	$1.05\ 10^{-5}$	$1.05\ 10^{-5}$
MSE(σ^2)	$1.19\ 10^{-6}$	$1.14\ 10^{-6}$	$1.12\ 10^{-6}$	$1.10\ 10^{-6}$	$1.13\ 10^{-6}$	$1.10\ 10^{-6}$	$1.09\ 10^{-6}$	$1.08\ 10^{-6}$
$\rho(1)$	0.0133	0.0096	0.0078	0.0053	$1.27\ 10^{-4}$	$-6.41\ 10^{-4}$	$-2.45\ 10^{-4}$	$-2.62\ 10^{-4}$
AR	1	1	1	1	0.9666	0.9769	0.9800	0.9832
m	71	109	121	177	71	109	121	177
Time	0.6849	0.6937	0.6957	0.7105	0.7339	0.7397	0.7380	0.7502

Table 8. Results of FUSS methods with different pruning procedures, $K = 5000$ and the Nakagami target with $\beta = 4.6$ and $\Omega = 1$.

	$\sigma_p= 0.2$	$\sigma_p= 0.5$	$\sigma_p= 0.8$	$\sigma_p= 1$	$\sigma_p= 2$	$\sigma_p= 3$	$\sigma_p= 4$
MH MSE(μ)	0.0021	$3.95\ 10^{-4}$	$1.98\ 10^{-4}$	$1.52\ 10^{-4}$	$1.43\ 10^{-4}$	$1.90\ 10^{-4}$	$2.52\ 10^{-4}$
MSE(σ^2)	0.0513	0.0091	0.0039	0.0027	$9.20\ 10^{-4}$	$6.18\ 10^{-4}$	$5.69\ 10^{-4}$
$\rho(1)$	0.8935	0.7495	0.7433	0.7611	0.8389	0.8808	0.9043
AR	1	1	1	1	1	1	1
Time	1	1	1	1	1	1	1
Slice sampling							
MSE(μ)= $1.24\ 10^{-5}$	MSE(σ^2)= $2.27\ 10^{-5}$		$\rho(1)$= 0.0229	AR=1		Time= 2.5037	

Table 9. Results of the standard MH and slice sampling methods, $K = 5000$ and the Nakagami target with $\beta = 4.6$ and $\Omega = 1$.

of iterations owing to the rejected samples, FUSS-RC is faster than the standard MH and the slice sampling. The time spent in FUSS-MH always increases (almost linearly) with m, whereas in FUSS-RC the computational cost can also decrease when m grows due to an improvement in the acceptance rate. where σ^2 is the variance of the target in Eq. (11). The pruning procedures P3 and P4 performs clearly better than P2. The best one, in terms of pruning performance, is P4, as expected. Indeed, P4 chooses the final support points in a better way: it can be seen comparing the time values, i.e., FUSS-RC-P4 is always faster than FUSS-RC-P3, even if more points are used. The reason is that the points are better located so that the L_1 distance between target and proposal is smaller and the acceptance rate greater. It can be noted observing also that FUSS-RC-P4 with $\delta = 0.9$ uses only $m = 71$ points and obtained AR= 0.9666, whereas FUSS-RC-P3 with $\delta = 0.5$ uses $m = 88$ achieving AR= 0.9630. Namely, FUSS-RC-P3 even with more points can have an AR smaller than FUSS-RC-P4. Thus, FUSS-RC-P4 results be faster than FUSS-RC-P3. On the other hand, FUSS-MH-P3 works lightly faster than FUSS-MH-P4, providing similar performance.

6.1.2. Multimodal target

Now we consider a multimodal target density, specifically a mixture of 6 Gaussian pdfs,

$$\bar{\pi}(x) = \sum_{i=1}^{4} \mathcal{N}(x; \mu_i, \sigma_i^2),$$

where $\mu_1 = -7$, $\sigma_1 = 0.1$, $\mu_2 = 0$, $\sigma_2 = 1$, $\mu_3 = 8$, $\sigma_3 = 0.2$, and $\mu_4 = 15$, $\sigma_4 = 0.1$. It clearly presents 4 modes as shown in Figure 2(a), and three of them are tight, so that this target becomes even more challenging. We consider first only $K = 200$ as total number of iterations of the chain, taking in account all the generated samples to make the estimation (without removing any burn-in period). We again test the performance of FUSS algorithm, the standard MH method with a Gaussian random walk proposal pdf and the slice sampling, as in the previous example. For the sake of simplicity we have considered only the FUSS-MH, in this case.[2] The initial state x_0 of the chain is chosen uniformly in $[-10, 20]$, i.e., $x_0 \in \mathcal{U}([-10, 20])$. As in the previous

---

[2]As shown in the previous numerical example, FUSS-RC provides in general better results wasting slightly more computational time.

example, we have used the Matlab functions directly provided by MathWorks (`mhsample.m` and `slicesample.m`). For FUSS-MH, we use an initial set $\mathcal{S}_M = \{-10^3, -10^3 + 0.01, \ldots, 10^3 - 0.01, 10^3\}$ then $M = 2\ 10^5 + 1$ points.

The MSEs in the estimation of the mean (μ) and variance (σ^2) of the target are shown in Table 10 for FUSS-MH and in Table 11 for the standard MH method, using $K = 200$ samples for both algorithms. We also provide the linear correlation at lag-1 (denoted as $\rho(1)$), the number of points after pruning (m) and the spent time normalized w.r.t. the time wasted by MH using $K = 200$. The results are averaged over $3 \cdot 10^4$ runs. Note that, in this case, the theoretical bound for the MSE$(\mu) \geq \frac{\sigma^2}{K}$ is $\frac{\sigma^2}{K} = \frac{68.765}{200} = 0.3438$, achievable via Monte Carlo using independent samples. Furthermore, in Table 12 we give the MSE obtained using MH and slice sampling, increasing the total number K of iterations of the Markov chain. Figures 2(b)-(c) depict respectively the log-MSE and log-time of MH and slice sampling as function of K.

Observing the results, it is apparent that FUSS methodology speeds up the convergence of the Markov chain, indeed:

- FUSS-MH always outperforms MH and slice sampling, providing an improvement at least of 41% in the MSE, and $\approx 98\%$ with the P4 pruning procedure.

- Table 11 also shows the importance of using an adaptive or self-tuned method: the results vary considerably with the choice of σ_p and, even in the best case, the MSE is much higher than in the worst case of FUSS-MH. This is owing to FUSS-MH is able to adapt the entire shape of the proposal pdf according to the target.

- FUSS-MH using the pruning P4 virtually reaches the theoretical bound for the MSE, i.e., 0.3438, obtainable using independent samples. This remarkable result is also achieved almost independently on the choice of the threshold δ.

- Observing Table 12 and Figures 2(b)-(c), we can see that in order to obtain the same performance of FUSS-MH using a standard MH method, we need to use between $K = 15000$ and $K = 20000$ iterations, wasting at least $+50\%$ more of computational time, comparing with FUSS-MH-P4 with $\delta = 0.9$. With the slice sampling we need to use between $K = 15000$ and $K = 20000$ iterations as well, speeding at least $+125\%$ more of computational time.

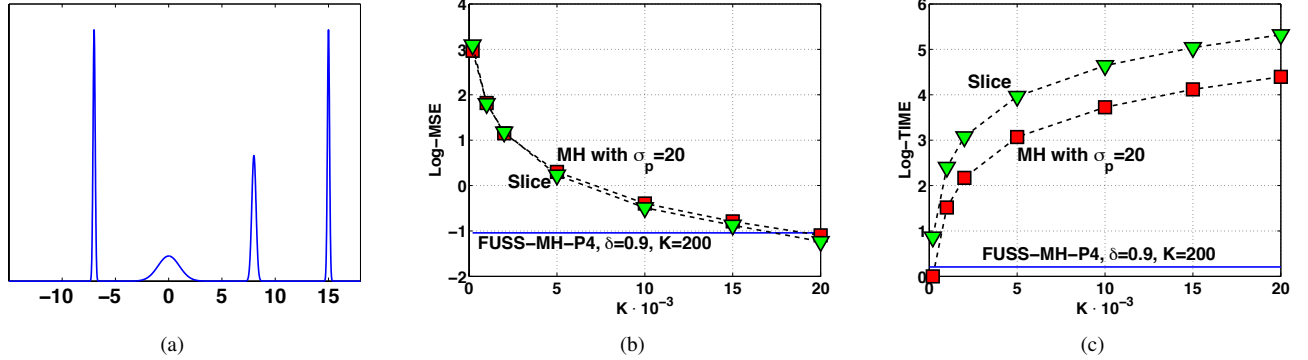|		**FUSS-MH**			
Pruning		$\delta = 0.9$	$\delta = 0.5$	$\delta = 0.3$	$\delta = 0.01$
P2	MSE(μ)	11.38	7.33	4.30	0.4680
	MSE(σ^2)	409.02	288.35	211.78	19.52
	$\rho(1)$	0.9142	0.8917	0.8419	0.1468
	m	18	46	103	662
	Time	1.03	1.04	1.06	1.17
P3	MSE(μ)	1.7683	1.1368	0.8686	0.3679
	MSE(σ^2)	78.96	49.82	35.92	15.3513
	$\rho(1)$	0.6693	0.5284	0.4224	0.0296
	m	61	99	135	497
	Time	1.20	1.27	1.31	1.35
P4	MSE(μ)	0.3786	0.3662	0.3638	0.3526
	MSE(σ^2)	15.53	15.31	15.10	14.53
	$\rho(1)$	0.0446	0.0306	0.0247	0.0093
	m	145	195	223	605
	Time	1.23	1.26	1.28	1.40

Table 10. Results of FUSS-MH with different pruning procedures and $K = 200$, drawing from the mixture of Gaussians target.

		$\sigma_p = 2$	$\sigma_p = 8$	$\sigma_p = 14$	$\sigma_p = 20$	$\sigma_p = 25$	$\sigma_p = 30$
MH	MSE(μ)	34.92	24.64	19.62	19.46	19.80	20.57
	MSE(σ^2)	4.43 10^3	1.36 10^3	1.13 10^3	1.09 10^3	1.37 10^3	1.51 10^3
	$\rho(1)$	0.7481	0.9465	0.9425	0.9424	0.9445	0.9459
	Time	1	1	1	1	1	1

Table 11. Results of the standard MH method with $K = 200$, drawing from the mixture of Gaussians target. The best results are obtained with $\sigma_p = 20$, but the MSE is always greater than using FUSS-MH.

		$K=200$	$K=10^3$	$K=2\cdot 10^3$	$K=5\cdot 10^3$	$K=10^4$	$K=1.5\cdot 10^4$	$K=2\cdot 10^4$
MH	MSE(μ)	19.46	6.16	3.16	1.35	0.6735	0.4540	0.3341
$\sigma_p=20$	MSE(σ^2)	1.09 10^3	263.36	123.37	44.69	22.48	14.45	10.79
	Time	1	4.55	8.75	21.51	41.48	61.44	81.06
Slice	MSE(μ)	22.13	6.08	3.25	1.26	0.6136	0.4177	0.2911
	MSE(σ^2)	1.10 10^3	171.13	68.91	23.98	10.48	7.40	5.85
	Time	2.37	11.04	21.62	52.66	103.78	154.38	204.57

Table 12. Results of the slice sampling and the standard MH method with $\sigma_p = 20$, varying the total number K of iterations of the chain, drawing from the mixture of Gaussians target.

Fig. 2. **(a)** The multimodal target pdf $\pi(x)$. **(b)** Log-MSE as function of $\frac{K}{10^3}$ for the slice (triangles) and the MH (squares) methods (with $\sigma_p = 20$). The solid line shows the log-MSE, $\log(0.3526) = -1.0424$, of FUSS-MH-P4 with $\delta = 0.9$, achieved using only $K = 200$. **(c)** The spent log-time as function of K for the slice (triangles) and the MH (squares) methods. The solid line corresponds to FUSS-MH-P4 with $\delta = 0.9$ and $K = 200$.

6.2. Toy example within Gibbs

In order to show the importance of the use of an adequate MCMC technique "within Gibbs", in this section we provide a simple example using different methods for drawing from the full-conditional pdfs: the results provided by the Gibbs sampler vary considerably depending on the technique used "within Gibbs". Let us consider the bivariate target density

$$\bar{\pi}(x_1, x_2) \propto \pi(x_1, x_2) = \exp\left(-\frac{(x^2 - A + By)^2}{4} - \frac{x^2}{2\sigma_1^2} - \frac{y^2}{2\sigma_2^2}\right),$$

with $A = 16$, $B = 10^{-2}$, and $\sigma_1^2 = \sigma_2^2 = \frac{10^4}{2}$. Densities of this analytic form have often been used in the literature [17, 9] to evaluate the performance of different Monte Carlo algorithms. We apply a Gibbs sampler to draw from $\bar{\pi}(x_1, x_2)$. To generate from the full conditional pdfs, we use FUSS-MH-P4 with $\delta = 0.9$, starting with a uniform grid $\mathcal{S}_M = \{-10^4, -10^4 + 0.01, \ldots, 10^4 - 0.01, 10^4\}$,i.e., $M = 2 \cdot 10^5 + 1$. In order to compare the performance, we also apply the standard ARMS method [6]: this technique uses an interpolation procedure to build the proposal using a set of support points, similarly to FUSS. However, unlike FUSS, the ARMS method starts with few support points and adds new ones adaptively. We choose for ARMS the initial set $\{-10, -6, -4.3, -0.01, 3.2, 3.8, 4.3, 7, 10\}$. Finally, we also consider a standard MH algorithm with a random walk proposal $\bar{\pi}(x_t|x_{t-1}) \propto \exp((x_t - x_{t-1})^2/(2\sigma_p^2))$ with $\sigma_p \in \{1, 2, 10\}$ (initial state $x_0 \in \mathcal{U}([-5, 5])$). We use $N_G = 2000$ iterations of the Gibbs sampler, using all the samples to estimate four statistics that involve the first four moments of the target: mean, variance, skewness and kurtosis. In each iteration, we draw K samples from each full-conditionals, take the last one and continue in the Gibbs cycle.

Table 6.2 provides the numerical results (averaged over 1000 runs) for the *Mean Absolute Error* (MAE) and the time required by the Gibbs sampler. The time is normalized by considering 1 to be the time elapsed by using ARMS within Gibbs and $K = 50$. Figure 3(a) illustrates the target $\bar{\pi}(\mathbf{x})$ whereas Figures 3(b)-(c) depict the MAE and the spent time of ARMS and FUSS-MH-P4 as function of K.

Observing Table 6.2 and Fig. 3(b), we notice that FUSS-MH-P4 outperforms ARMS for all values of K in the estimations of the four central moments: FUSS-MH-P4 achieves always MAEs close to zero. As shown also in Fig. 3(c), due to the pruning procedure, FUSS-MH-P4 is a slightly slower than ARMS for $K = 3, 5, 10$. However, the computational time FUSS-MH-P4

| Technique | K | MAE | | | | Time |
		Mean	Variance	Skewness	Kurtosis	
FUSS-MH-P4 $\delta = 0.9$	3	0.0735	0.0365	0.0369	0.0022	0.343
	5	0.0735	0.0361	0.0367	0.0022	0.343
	10	0.0724	0.0354	0.0365	0.0021	0.345
	50	0.0721	0.0355	0.0364	0.0021	0.348
ARMS	3	3.408	11.580	3.384	11.572	0.077
	5	3.151	9.839	2.650	7.079	0.116
	10	2.798	7.665	2.024	4.124	0.223
	50	1.918	3.407	1.134	1.292	1.000
MH $\sigma_p = 1$	100	3.509	12.308	3.671	13.666	0.540
$\sigma_p = 2$	100	1.756	3.077	0.9782	0.9633	0.540
$\sigma_p = 10$	100	0.0756	0.0376	0.0368	0.0025	0.540
$\sigma_p = 1$	1000	3.508	12.302	3.665	13.624	3.229
$\sigma_p = 2$	1000	1.601	2.560	0.8741	0.7691	3.229
$\sigma_p = 10$	1000	0.0743	0.0360	0.0363	0.0021	3.229

Table 13. MAE in the estimation of four statistics (first component) of $\bar{\pi}(x_1, x_2)$ and normalized simulation time. All the techniques are used within a Gibbs sampler ($N_G = 2000$ total iterations), performing K iterations for each full-conditionals.

remains virtually constant with K whereas in ARMS it increases with K, since ARMS adds new support points for improving the proposal pdf, hence becoming more costly. Regarding the use of the MH algorithm within Gibbs, the results depend largely on the choice of the variance of the proposal, σ_p^2, showing the needed of adaptive or self-tuned MCMC strategies. Indeed, for an inadequate scale parameter (e.g., $\sigma_p = 1$ or $\sigma_p = 2$), even the use of $K = 1000$ provides bad results. On the other hand, when a good σ_p is selected (i.e., $\sigma_p = 10$), MH with $K = 100$ and $K = 1000$ provides virtually the same performance of FUSS-MH-P4 but obviously wasting more computational cost.

(a)　　　　　(b)　　　　　(c)
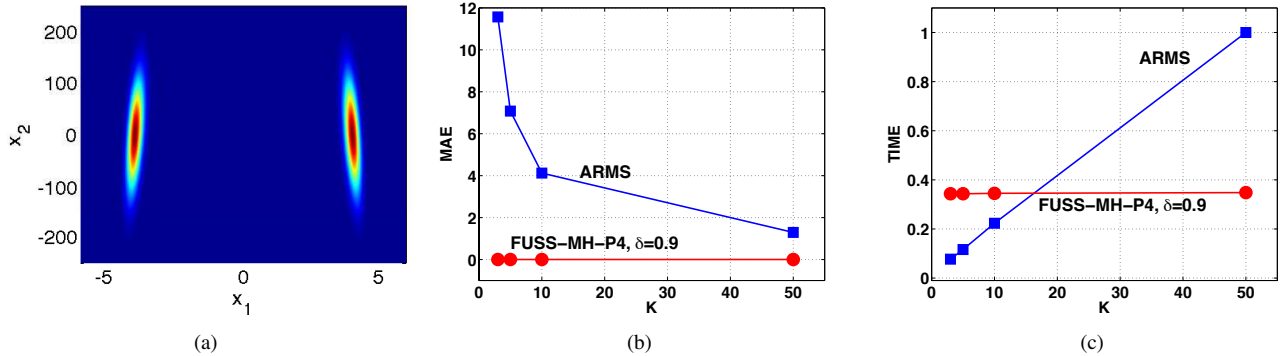
Fig. 3. **(a)** Contour plot of the target pdf $\bar{\pi}(x_1, x_2)$. **(b)** MAE in estimation of the kurtosis (first component) as function of K, using ARMS-within-Gibbs (squares) and FUSS-within-Gibbs (circles). **(c)** Normalized spent time as function of K for ARMS (squares) and FUSS (circles) within the Gibbs sampler.

7. CONCLUSIONS

In this work, we have introduced a novel extremely efficient MCMC sampler for drawing from univariate densities. The achieved performance are close to an exact sampler yielding i.i.d. samples. The novel technique outperforms (in terms of performance and speed) other wide-used MCMC methodologies as the Metropolis-Hastings (MH) method, the slice sampling and the ARMS algorithm. The time comparison has been realized using their MATLAB MathWorks implementation (i.e., `mhsample.m` and `slicesample.m`). Numerical simulations also show the clear advantage of use the FUSS algorithm within a Gibbs sampler. Moreover, although the performance are remarkable, the new technique is *not* an adaptive MCMC, hence there are not issues with the ergodicity.

8. ACKNOWLEDGEMENTS

References

[1] N. C. Beaulieu and C. Cheng. Efficient Nakagami-m fading channel simulation. *IEEE Transactions on Vehicular Technology*, 54(2):413–424, 2005.

[2] P. M. Djurić, J. H. Kotecha, J. Zhang, Y. Huang, T. Ghirmai, M. F. Bugallo, and J. Míguez. Particle filtering. *IEEE Signal Processing Magazine*, 20(5):19–38, September 2003.

[3] A. Doucet and X. Wang. Monte Carlo methods for signal processing. *IEEE Signal Processing Magazine*, 22(6):152–170, Nov. 2005.

[4] W. J. Fitzgerald. Markov chain Monte Carlo methods with applications to signal processing. *Signal Processing*, 81(1):3–18, January 2001.

[5] W. R. Gilks. Derivative-free Adaptive Rejection Sampling for Gibbs Sampling. *Bayesian Statistics*, 4:641–649, 1992.

[6] W. R. Gilks, N. G. Best, and K. K. C. Tan. Adaptive Rejection Metropolis Sampling within Gibbs Sampling. *Applied Statistics*, 44(4):455–472, 1995.

[7] W. R. Gilks, R.M. Neal, N. G. Best, and K. K. C. Tan. Corrigidum: Adaptive Rejection Metropolis Sampling within Gibbs Sampling. *Applied Statistics*, 46(4):541–542, 1997.

[8] W. R. Gilks and P. Wild. Adaptive Rejection Sampling for Gibbs Sampling. *Applied Statistics*, 41(2):337–348, 1992.

[9] H. Haario, E. Saksman, and J. Tamminen. Adaptive proposal distribution for random walk Metropolis algorithm. *Computational Statistics*, 14:375–395, 1999.

[10] L. Holden, R. Hauge, and M. Holden. Adaptive independent Metropolis-Hastings. *The Annals of Applied Probability*, 19(1):395–413, 2009.

[11] J. M. Keith, D. P. Kroese, and G. Y. Sofronov. Adaptive independence samplers. *Statistics and Computing*, 18(4):409–420, 2008.

[12] J. Kotecha and Petar M. Djurić. Gibbs sampling approach for generation of truncated multivariate Gaussian random variables. *Proceedings of Acoustics, Speech, and Signal Processing, (ICASSP)*, 1999.

[13] F. Liang, C. Liu, and R. Caroll. *Advanced Markov Chain Monte Carlo Methods: Learning from Past Samples*. Wiley Series in Computational Statistics, England, 2010.

[14] J. S. Liu. *Monte Carlo Strategies in Scientific Computing*. Springer, 2004.

[15] D. Luengo and L. Martino. Almost rejectionless sampling from Nakagami-m distributions ($m \geq 1$). *IET Electronics Letters*, 48(24):1559–1561, 2012.

[16] L. Martino, R. Casarin, F. Leisen, and D. Luengo. Adaptive sticky generalized Metropolis. *arXiv:1308.3779*, 2013.

[17] L. Martino and J. Read. On the flexibility of the design of multiple try Metropolis schemes. *arXiv:1201.0646 (submitted to Computational Statistics)*, January 2012.

[18] L. Martino, J. Read, and D. luengo. Independent doubly adaptive rejection Metropolis sampling. In *Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2014.

[19] R. Meyer, B. Cai, and F. Perron. Adaptive rejection Metropolis sampling using Lagrange interpolation polynomials of degree 2. *Computational Statistics and Data Analysis*, 52(7):3408–3423, March 2008.

[20] K. P. Murphy. *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012.

[21] S. Olver and A. Townsend. Fast inverse transform sampling in one and two dimensions. *arXiv:1307.1223*, 2013.

[22] J. G. Proakis. *Digital Communications*. McGraw-Hill, Singapore, 1995.

[23] C. P. Robert and G. Casella. *Monte Carlo Statistical Methods*. Springer, 2004.

[24] J. K. O. Ruanaidh and W. J. Fitzgerald. *Numerical Bayesian Methods Applied to Signal Processing*. Springer, 1996.

[25] W. Shao, G. Guo, F. Meng, and S. Jia. An efficient proposal distribution for Metropolis-Hastings using a b-splines technique. *Computational Statistics and Data Analysis*, 53:465–478, 2013.

[26] L. Tierney. Exploring posterior distributions using Markov Chains. *Computer Science and Statistics: Proceedings of IEEE 23rd Symp. Interface*, pages 563–570, 1991.

[27] L. Tierney. Markov chains for exploring posterior distributions. *The Annals of Statistics*, 22(4):1701–1728, 1994.

A. TAILS

The choice of the tails is important for two reasons: (a) to accelerate the convergence of the chain to the target (in the case, for instance, of heavy-tailed target distributions) and (b) to increase the robustness of the method w.r.t. the initial choice of the set $\mathcal{S}_M$.

In general, the construction of tails with a bigger area below them reduces the dependence on a specific choice of the initial support points. When it is impossible, a good choice is to build tails such that $p(x) \geq \pi(x)$ for $x \in \mathcal{I}_0$ and $x \in \mathcal{I}_m$. This is always possible when the target pdf has light tails (i.e., convex tails in the log-domain) as we have shown in Figure 1 and explained below.

A.0.1. Light Tails

In this case, we use two exponential pieces

$$p(x|\mathcal{S}_m) = e^{h_0 x + b_0}, \ \ \forall x \in \mathcal{I}_0; \quad p(x|\mathcal{S}_m) = e^{h_m x + b_m}, \ \ \forall x \in \mathcal{I}_m.$$

The linear function, $w_0(x) = h_0 x + b_0$ is the straight line passing through the points $(s_1, V(s_1))$ and $(s_2, V(s_2))$, whereas the second linear function $w_m(x) = h_m x + b_m$ is the straight line passing through $(s_{m-1}, V(s_{m-1}))$ and $(s_m, V(s_m))$. Note that we can easily compute analytically the area below each piece, and we can also easily draw from each exponential tail by inversion method [23].

A.1. Heavy Tails

For heavy tails, there are also several possibilities. For instance, here we propose to use Pareto pieces with the analytic form

$$p(x|\mathcal{S}_m) = e^{\rho_0} \frac{1}{|x - \mu_0|^{\gamma_0}}, \ \ \forall x \in \mathcal{I}_0; \quad p(x|\mathcal{S}_m) = e^{\rho_m} \frac{1}{|x - \mu_m|^{\gamma_m}}, \ \ \forall x \in \mathcal{I}_m,$$

directly in the pdf domain with $\gamma_j > 1$, $j \in \{0, m\}$. In the log-domain, this is equivalent to

$$w_0(x) = \rho_0 - \gamma_0 \log(|x - \mu_0|), \quad \text{for} \quad x \in \mathcal{I}_0,$$
$$w_m(x) = \rho_m - \gamma_m \log(|x - \mu_m|), \quad \text{for} \quad x \in \mathcal{I}_m,$$

Fixed the parameters μ_j, $j \in \{0, m\}$, the remaining ρ_j and γ_j are set in order to satisfying the passing conditions through the points $(s_1, V(s_1))$, $(s_2, V(s_2))$, and through $(s_{m-1}, V(s_{m-1}))$, $(s_m, V(s_m))$, respectively. The parameters μ_j can be arbitrarily chosen by the user, fulfilling the inequalities

$$\mu_0 > s_2, \quad \mu_m < s_{m-1}.$$

Values of μ_j such that $\mu_0 \approx s_2$ and $\mu_m \approx s_{m-1}$ yields small values of γ_j (close to 1) and, as a consequence, fatter tails. Greater differences $|\mu_0 - s_2|$ and $|\mu_m - s_{m-1}|$ yields $\gamma_j \to +\infty$, i.e., lighter tails. In the limit case, they coincide with the exponential construction presented above. Different examples of construction are illustrated in Figure 4.

As in the previous case, we can compute analytically the integral of $p(x|\mathcal{S}_m)$ in $\mathcal{I}_0$ and $\mathcal{I}_m$. We can also easily draw from each Pareto tail by inversion method [23].
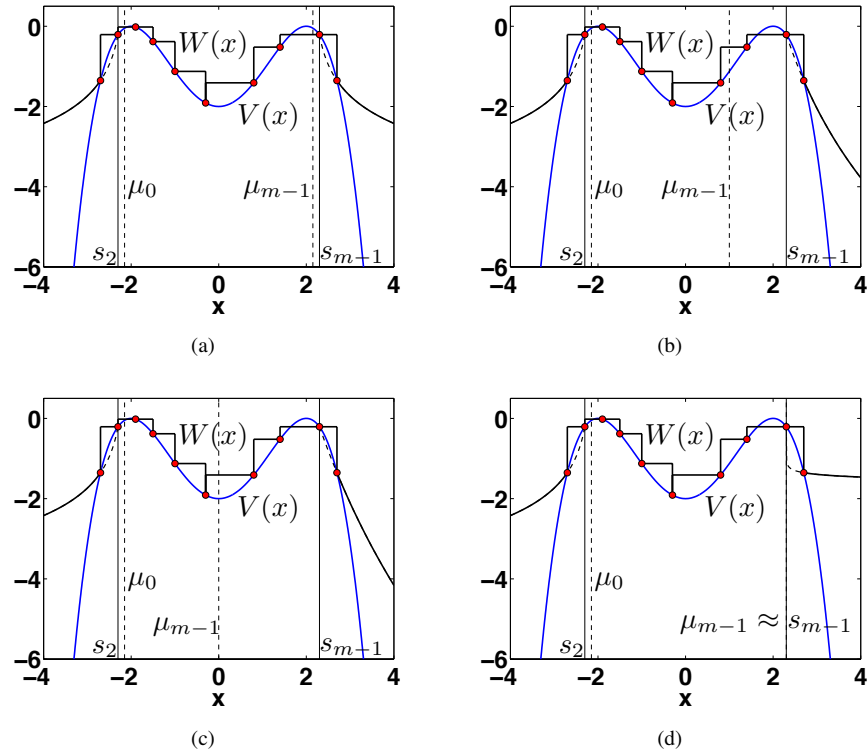
Fig. 4. Example of construction of heavy tails in the log-domain of the pdf. **(a)** Example with a specific choice of the parameters $\mu_0 > s_2$ and $\mu_{m-1} < s_{m-1}$ ($m = 10$ in figure). **(b)-(c)** Alternative right log-tail constructions, decreasing the parameter μ_{m-1}. For $\mu_{m-1} \to -\infty$, the log-tail tends to be a straight line passing through $\left(s_{m-1}, V(s_{m-1})\right)$, $\left(s_m, V(s_m)\right)$. **(d)** Construction of the right log-tail with $\mu_{m-1} \approx s_{m-1}$. In this case, $\mu_{m-1} \to s_{m-1}$, it tends to be a constant line.