# 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]University of Helsinki,
[2]Tampere University of Technology,
[3]Universidad Politecnica de Madrid.

Please note: this is only a preliminary draft.

**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 [12, 19] 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 [11, 12]. The two best known MCMC approaches are the Metropolis-Hastings (MH) algorithm and the Gibbs sampler [19]. The Gibbs sampling technique is extensively used in Bayesian inference [19], e.g., in signal processing [20] and machine learning [17], 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 MCMC-inside-another-MCMC approach. 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 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], *Independent Doubly Adaptive Rejection Metropolis Sampling* (IA2RMS) [16], and *Adaptive Sticky Metropolis* [14] 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, 19].

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 (particularly with a MATLAB implementation) and extremely efficient (it yields almost independent samples), as shown in the numerical results, even with high 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 and spiky target densities, i.e., densities with several sharp modes, where virtually all of the existent MCMC techniques often fail.

## 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 [19]. 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 [19, 3, 10], 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 (as we show in the next section). The general structure is given in table 1. We consider here two possible techniques for the step 4 of FUSS:

- *The Metropolis-Hastings (MH) algorithm [19]:* it is shown in Table 2. In this case, we denote the whole method as *FUSS-MH*.
- *The Rejection chain algorithm [21, 22]:* 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 whole 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 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 $\bar{\pi}(x)$. Note that, in this scenario, the probability $\alpha_{RC}$ of accepting the movement is always 1.

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 $1 - \alpha$ set $x_{k+1} = x_k$.

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

- If FUSS-RC, after the rejection step, the "true" proposal pdf is

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

Thus, $q(x)$ is used as proposal in the acceptance function $\alpha_{RC}$. 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

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

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

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

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

$V(x) = \log[\pi(x)]$ and $W(x) = \log[p(x|\mathcal{S}_m)]$. In FUSS-MH, the acceptance probability can be expressed as

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

the ratio in the RS test becomes $e^{W(x')-V(x')}$ 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. If the final built proposal pdf 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, the proposal $p(x|\mathcal{S}_m)$ is built according to the new set $\mathcal{S}_m$. After that, the proposal $p$ is kept fixed. Thus, the FUSS techniques are standard non-adaptive MCMC, avoiding any issue about the ergodicity.

## 4 Initialization and general FUSS strategy

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.,

  $$\mathcal{S}_M = s_1, s_2 = s_1 + \epsilon, s_3 = s_1 + 2\epsilon, \ldots, s_M,$$

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

- Reduce the number of support points according to a certain criterion (see Section 6 for some examples).

- Build a stepwise approximation of the target pdf given the pruned support points (see Section 5).

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).

## 5 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 $j = 1, ..., m_t - 1$ 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,

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

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 log-tail. For instance, choosing light tails, $w_j(x)$, $j = \{0, m\}$, are linear functions. For further details, see Appendix A and Figures 1-2.

The choice of taking the maximum is to satisfy, in more regions as possible, the inequality $W(x) \geq V(x)$. If this inequality is verified for all $x \in \mathcal{X}$ then also $p(x|\mathcal{S}_m) \geq \pi(x)$, $\forall x \in \mathcal{X}$, so that the proposed algorithm becomes a standard rejection sampler, providing independent samples.

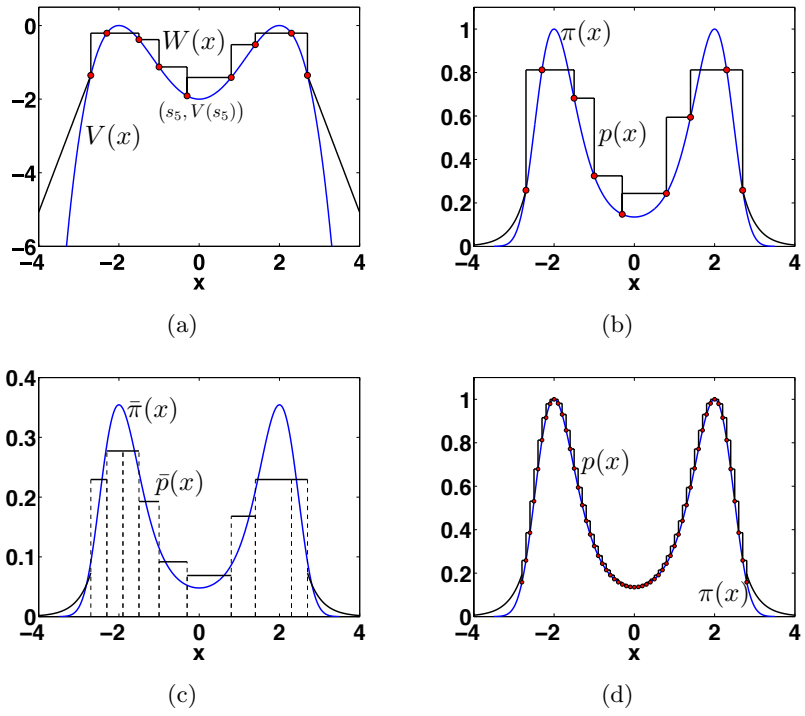


Figure 1: Example of proposal construction. **(a)** Construction procedure with $m = 9$ support points, in the log-domain. The log-tails are in this case light tails (two straight lines). **(b)** The corresponding unnormalized densities $p(x) = e^{W(x)}$ and $\pi(x) = e^{V(x)}$. **(c)** The corresponding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

### 5.1 Variate generation from $\bar{p}(x)$

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

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

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

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

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 (exponential or Pareto) as well. Moreover, normalize them

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

2. Choose a piece $j^* \in \{0, \ldots, m\}$ according to the normalized weights $\eta_i$, $i = 0, \ldots, m$.

3. Draw $x' \sim \phi_{j^*}(x) \propto \exp\left(w_{j^*}(x)\right)$.

It is important to remark that the process of calculate the areas $A_i$ and then the weights $\eta_i$ is done once, before running the Markov chain. The calculation of the areas $A_i$ of the rectangular regions is straightforward and fast (as also the tails).

## 6 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

Table 4: **Pruning algorithm P1**

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 decreasing order
$$\pi(s_{r_1}) = \max_{1 \leq i \geq N} \pi(s_i) \geq \pi(s_{r_2}) \geq .... \geq \pi(s_{r_M}) = \min_{1 \leq i \geq N} \pi(s_i).$$
3. Return $\mathcal{S}_m = \{s_{r_1}, s_{r_2}, \ldots, s_{r_m}\}$.

Table 5: **Pruning algorithm P2**

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

simple idea of pruning all the points $s_r$ with "small" value $\pi(s_r)$ (smaller than the others). They are the simplest and fastest ones but they must use more carefully than the rest. Moreover, 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.

A more refined pruning technique P3 in Table 6 can be easily provided. The underlying idea is that we can remove the support points between which 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 $\pi$. 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 $\mathcal{S}_m = \mathcal{S}^{(n)} \setminus \{s_1\}$.

## 6.1 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 [19],

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

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 distance within the $i$-th interval ($0 \leq j \leq m$),

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

where for simplicity we define $s_0 = -\infty$ and $s_{m+1} = +\infty$, i.e., $\mathcal{I}_0 = (-\infty, s_1]$ and $\mathcal{I}_m = [s_m, \infty)$. 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 $\pi(x)$.

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
$$L = \max_{1 \le j \le \left\lfloor \frac{m-1}{2} \right\rfloor} (s_{2j+1} - s_{2j-1}) |\pi(s_{2j+1}) - \pi(s_{2j-1})|.$$
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 $b_r \le \delta L$, 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 $\mathcal{S}_m = \mathcal{S}^{(R)}$.

Since, in this work, the proposal $p(x)$ is a piecewise constant function (with the exception of the tails) where each constant piece is $\exp(w_i) = \exp(\max[V(s_i), V(s_{i+1})])$, and considering a continuous target pdf, it is apparent that

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

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 base of the pruning strategy detalied 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 $R = \left\lfloor \frac{M-1}{2} \right\rfloor$ points can be removed. The procedure is iterated until no more points are pruned, i.e., when all the bounds $b_r$ are greater than the threshold $\delta$. Clearly, the algorithm could be stopped before.

# 7 Simulations

## 7.1 Unimodal pdf: Nakami target

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, 13, 18]. When $\beta$ 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 $\beta$ 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 $\delta$. 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 [12, 19] 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 $\sigma_p$. Finally, we consider another well-known methodology the *slice sampling* technique [19, 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 $\mu$ and $\sigma^2$, the acceptance rate ($0 \leq$AR$\leq 1$) in the rejection sampling (RS) step, the number of points $m$ after the pruning and the wasted time. The time values are normalized w.r.t. the time wasted in the MH method. Due to only FUSS-RC has an RS step, in the other cases, AR is considered 1 since no sample is 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

| | | FUSS-MH | | | | FUSS-RC | | | |
|---|---|---|---|---|---|---|---|---|---|
| **Pruning** | | **$\delta$= 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($\mu$) | 1.1466 | 0.4067 | 0.0501 | $1.09\ 10^{-5}$ | 1.1621 | 0.4444 | 0.0513 | $1.05\ 10^{-5}$ |
| | MSE($\sigma^2$) | 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($\mu$) | 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($\sigma^2$) | $6.51\ 10^{-4}$ | $5.94\ 10^{-5}$ | $6.11\ 10^{-5}$ | $1.13\ 10^{-6}$ | $3.02\ 10^{-5}$ | $1.09\ 10^{-6}$ | $8.56\ 10^{-5}$ | $1.10\ 10^{-6}$ |
| | $\rho(1)$ | 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 | 0.9730 | 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($\mu$) | $1.10\ 10^{-5}$ | $1.09\ 10^{-5}$ | $1.06\ 10^{-5}$ | $1.06\ 10^{-5}$ | $1.10\ 10^{-5}$ | $1.09\ 10^{-5}$ | $1.05\ 10^{-5}$ | $1.05\ 10^{-5}$ |
| | MSE($\sigma^2$) | $1.19\ 10^{-6}$ | $1.14\ 10^{-6}$ | $1.12\ 10^{-6}$ | $1.10\ 10^{-6}$ | $1.13\ 10^{-6}$ | $1.10\ 10^{-6}$ | $1.09\ 10^{-6}$ | $1.08\ 10^{-6}$ |
| | $\rho(1)$ | 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($\mu$) | 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($\sigma^2$) | 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($\mu$)= $1.24\ 10^{-5}$ | | MSE($\sigma^2$)= $2.27\ 10^{-5}$ | | $\rho(1)$= 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 $\Omega = 1$.

virtually reaches the performance of an *exact sampler* in the estimation of $\mu$ 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,

[1]FUSS-RC always has less correlations among samples than FUSS-MH and, as a

in spite of the greater number 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 $\sigma^2$ 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= 0.9668, whereas FUSS-RC-P3 with $\delta = 0.5$ uses $m = 88$ achieving AR= 0.9577 and FUSS-RC-P3 with $\delta = 0.3$ uses $m = 106$ obtaining AR= 0.9635. 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.

## 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.


consequence, less variance in the estimation, but in some cases it can have more bias.

[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] 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.

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

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

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

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

[15] 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.

[16] 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.

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

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

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

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

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

[22] 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 avoiding "to cut-off" some modes, which are not contained in the thin grid $\mathcal{S}_M$, for instance.

In general, the construction of tails with a great 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).

### A.0.1 Light Tails

In this case, we use two exponential pieces

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

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 [19].

### A.1 Heavy Tails

For heavy tails, we propose to use Pareto pieces with the analytic form

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

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

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

Fixed the parameters $\mu_j$, $j \in \{0, m\}$, the remaining $\rho_j$ and $\gamma_j$ 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 $\mu_j$ can be arbitrarily chosen by the user, fulfilling the inequalities

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

Values of $\mu_j$ such that $\mu_0 \approx s_2$ and $\mu_m \approx s_{m-1}$ yields small values of $\gamma_j$ (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 2.

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 [19].

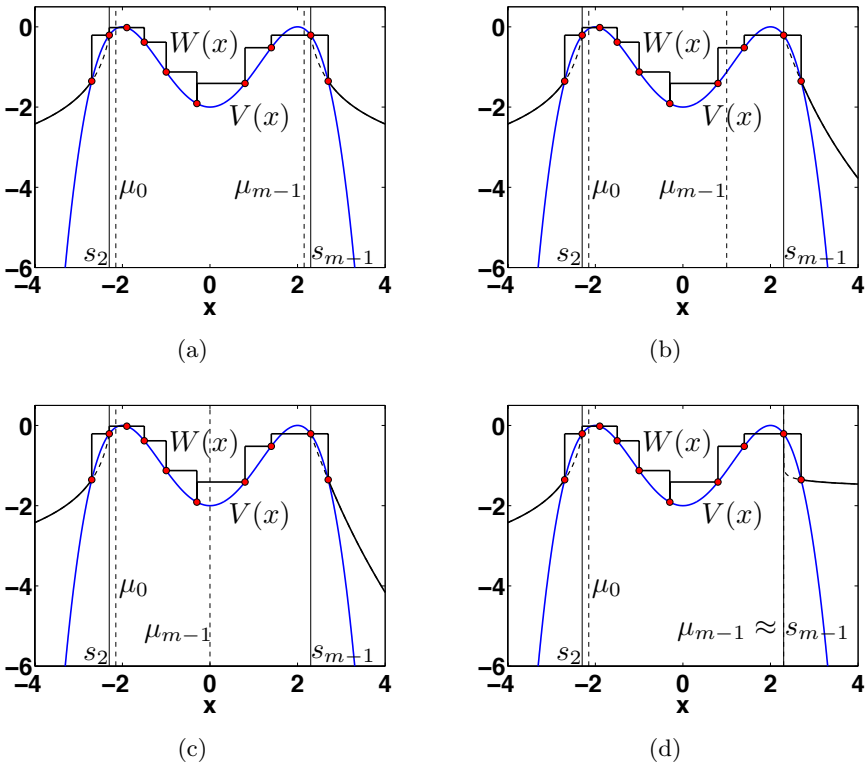


Figure 2: Example of construction of heavy tails in the log-domain of the pdf. **(a)** Example with a specific choice of the parameters $\mu_0 > s_2$ and $\mu_{m-1} < s_{m-1}$ ($m = 10$ in figure). **(b)-(c)** Alternative right log-tail constructions, decreasing the parameter $\mu_{m-1}$. For $\mu_{m-1} \to -\infty$, the log-tail tends to be a straight line passing through $(s_{m-1}, V(s_{m-1}))$, $(s_m, V(s_m))$. **(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.