The Bouncy Particle Sampler: A Non-Reversible Rejection-Free Markov Chain Monte Carlo Method

Alexandre Bouchard-Côté*, Sebastian J. Vollmer† and Arnaud Doucet‡

November 27, 2018

*Department of Statistics, University of British Columbia, Canada.

†Mathematics Institute and Department of Statistics, University of Warwick, UK.

‡Department of Statistics, University of Oxford, UK.

Abstract

Many Markov chain Monte Carlo techniques currently available rely on discrete-time reversible Markov processes whose transition kernels are variations of the Metropolis–Hastings algorithm. We explore and generalize an alternative scheme recently introduced in the physics literature [22] where the target distribution is explored using a continuous-time non-reversible piecewise-deterministic Markov process. In the Metropolis–Hastings algorithm, a trial move to a region of lower target density, equivalently of higher "energy", than the current state can be rejected with positive probability. In this alternative approach, a particle moves along straight lines around the space and, when facing a high energy barrier, it is not rejected but its path is modified by bouncing against this barrier. By reformulating this algorithm using inhomogeneous Poisson processes, we exploit standard sampling techniques to simulate exactly this Markov process in a wide range of scenarios of interest. Additionally, when the target distribution is given by a product of factors dependent only on subsets of the state variables, such as the posterior distribution associated with a probabilistic graphical model, this method can be modified to take advantage of this structure by allowing computationally cheaper "local" bounces which only involve the state variables associated to a factor, while the other state variables keep on evolving. In this context, by leveraging techniques from chemical kinetics, we propose several computationally efficient implementations. Experimentally, this new class of Markov chain Monte Carlo schemes compares favorably to state-of-the-art methods on various Bayesian inference tasks, including for high dimensional models and large data sets.

Keywords: Inhomogeneous Poisson process; Markov chain Monte Carlo; Piecewise deterministic Markov process; Probabilistic graphical models; Rejection-free simulation.

1 Introduction

Markov chain Monte Carlo (MCMC) methods are standard tools to sample from complex high-dimensional probability measures. Many MCMC schemes available at present are based on the Metropolis-Hastings (MH) algorithm and their efficiency is strongly dependent on the ability of the user to design proposal distributions capturing the main features of the target distribution; see [16] for a comprehensive review. We examine, analyze and generalize here a different approach to sample from distributions on $\mathbb{R}^d$ that has been recently proposed in the physics literature [22]. Let the energy be defined as minus the logarithm of an unnormalized version of the target density.

In this methodology, a particle explores the space by moving along straight lines and, when it faces a high energy barrier, it bounces against the contour lines of this energy. This non-reversible rejection-free MCMC method will be henceforth referred to as the Bouncy Particle Sampler (BPS). This algorithm and closely related schemes have already been adopted to simulate complex physical systems such as hard spheres, polymers and spin models [15, 18, 19, 21]. For these models, it has been demonstrated experimentally that such methods can outperform state-of-the-art MCMC methods by up to several orders of magnitude.

However, the implementation of the BPS proposed in [22] is not applicable to most target distributions arising in statistics. In this article we make the following contributions:

Simulation schemes based on inhomogeneous Poisson processes: by reformulating explicitly the bounces times of the BPS as the first arrival times of inhomogeneous Poisson Processes (PP), we leverage standard sampling techniques [5, Chapter 6] and methods from chemical kinetics [26] to obtain new computationally efficient ways to simulate the BPS process for a large class of target distributions.

Factor graphs: when the target distribution can be expressed as a factor graph [27], a representation generalizing graphical models where the target is given by a product of factors and each factor can be a function of only a subset of variables, we adapt a physical multi-particle system method discussed in [22, Section III] to achieve additional computational efficiency. This local version of the BPS only manipulates a restricted subset of the state components at each bounce but results in a change of all state components, not just the one being updated contrary the Gibbs sampler.

Ergodicity analysis: we present a proof of the ergodicity of BPS when the velocity of the particle is additionally refreshed at the arrival times of an homogeneous PP. When this refreshment step is not carried out, we exhibit a counter-example where ergodicity does not hold.

Efficient refreshment: we propose alternative refreshment schemes and compare their computational efficiency experimentally.

Empirically, these new MCMC schemes compare favorably to state-of-the-art MCMC methods on various Bayesian inference problems, including for high-dimensional scenarios and large data sets. Several additional original extensions of the BPS including versions of the algorithm which are applicable to mixed continuous-discrete distributions, distributions restricted to a compact support and a method relying on the use of curved dynamics instead of straight lines can be found in [**?**]. For brevity, these are not discussed here.

The rest of this article is organized as follows. In Section 2, we introduce the basic version of the BPS, propose original ways to implement it and prove its ergodicity under weak assumptions. Section 3 presents a modification of the basic BPS which exploits a factor graph representation of the target distribution and develops computationally efficient implementations of this scheme. In Section 4, we demonstrate this methodology on various Bayesian models. The proofs are given in the Appendix and the Supplementary Material.

2 The bouncy particle sampler

2.1 Problem statement and notation

Consider a probability distribution π on $\mathbb{R}^d$, equipped with the Borel σ-algebra $\mathcal{B}(\mathbb{R}^d)$. We assume that π admits a probability density with respect to the Lebesgue measure $\mathrm{d}x$ and slightly abuse notation by denoting also this density by π. In most practical scenarios, we only have access to an unnormalized version of this density, that is

$$\pi(x) = \frac{\gamma(x)}{\mathcal{Z}},$$

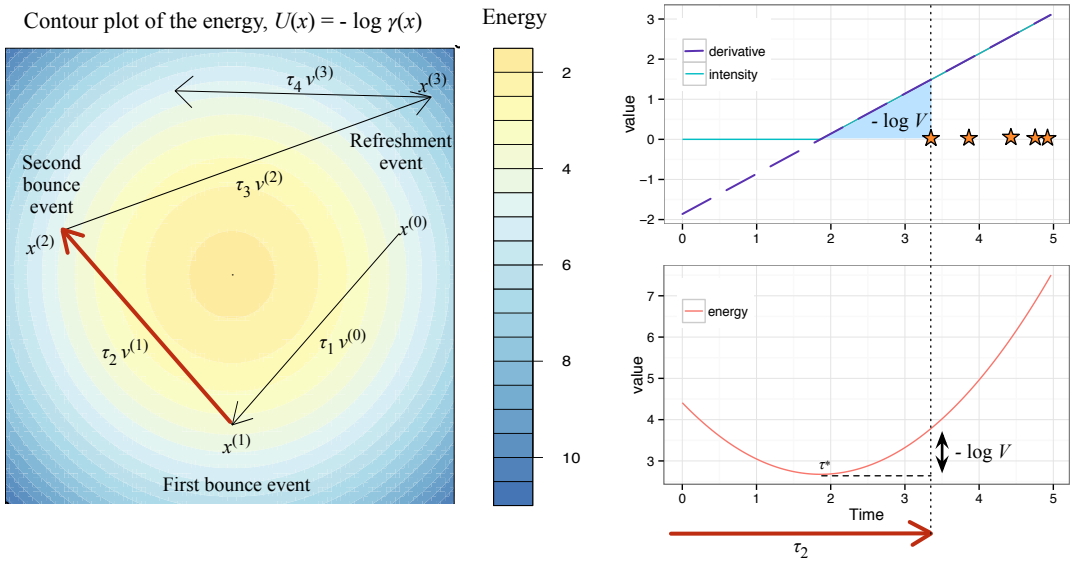

Figure 1: Illustration of BPS on a standard bivariate Gaussian distribution. Left and top right: see Section 2.2; bottom right: see Example 1.

where $\gamma : \mathbb{R}^d \to (0, \infty)$ can be evaluated pointwise but the normalizing constant $\mathcal{Z} = \int_{\mathbb{R}^d} \gamma(x)\,\mathrm{d}x$ is unknown. We call

$$U(x) = -\log \gamma(x)$$

the associated energy, which is assumed continuously differentiable, and we denote by $\nabla U(x) = \left(\frac{\partial U(x)}{\partial x_1}, \ldots, \frac{\partial U(x)}{\partial x_d}\right)^\top$ the gradient of U evaluated at x. We are interested in approximating numerically the expectation of arbitrary test functions $\varphi : \mathbb{R}^d \to \mathbb{R}$ with respect to π.

2.2 Algorithm description

The BPS methodology introduced in [22] simulates a continuous piecewise linear trajectory $\{x(t)\}_{t\geq 0}$ in $\mathbb{R}^d$. It has been informally derived as a continuous-time limit of the Metropolis algorithm in [22]. Each segment in the trajectory is specified by an initial position $x^{(i)} \in \mathbb{R}^d$, a length $\tau_{i+1} \in \mathbb{R}^+$ and a velocity $v^{(i)} \in \mathbb{R}^d$ (example shown in Figure 1, left). We denote the times where the velocity changes by $t_i = \sum_{j=1}^{i} \tau_j$ for $i \geq 1$, and set $t_0 = 0$ for convenience. The position at time $t \in [t_i, t_{i+1})$ is thus interpolated linearly, $x(t) = x^{(i)} + v^{(i)}(t - t_i)$, and each segment is connected to the next, $x^{(i+1)} = x^{(i)} + v^{(i)}\tau_{i+1}$. The length of these segments is governed by an inhomogeneous PP of intensity function $\lambda : \mathbb{R}^d \times \mathbb{R}^d \to [0, \infty)$

$$\lambda(x, v) = \max\{0, \langle \nabla U(x), v\rangle\}. \tag{1}$$

When the particle bounces, its velocity is updated in the same way as a Newtonian elastic collision on the hyperplane tangential to the gradient of the energy. Formally, the velocity after bouncing is given by

$$R(x)v = \left(I_d - 2\frac{\nabla U(x)\{\nabla U(x)\}^\top}{\|\nabla U(x)\|^2}\right) v = v - 2\frac{\langle \nabla U(x), v\rangle}{\|\nabla U(x)\|^2}\nabla U(x), \tag{2}$$

where I_d denotes the $d \times d$ identity matrix, $\|\cdot\|$ the Euclidean norm, and $\langle w, z\rangle = w^t z$ the scalar product between column vectors w, z. [1][22] also refresh the velocity at periodic times. We slightly

[1] Computations of the form $R(x)v$ are implemented via the right-hand side of Equation 2 which takes time $O(d)$ rather than the left-hand side, which would take time $O(d^2)$.

modify their approach by performing a velocity refreshment at the arrival times of a homogeneous PP of intensity $\lambda^{\text{ref}} \geq 0$, λ^{ref} being a parameter of the algorithm. A similar refreshment scheme was used for a related process in [18]. Throughout the paper, we use the terminology "event" for a time at which either a bounce or a refreshment occurs. The basic version of the BPS algorithm proceeds as follows:

Algorithm 1 Basic BPS algorithm

1. Initialize $\left(x^{(0)}, v^{(0)}\right)$ arbitrarily on $\mathbb{R}^d \times \mathbb{R}^d$ and let T denote the requested trajectory length.
2. For $i = 1, 2, \ldots$
   (a) Simulate the first arrival time $\tau_{\text{bounce}} \in (0, \infty)$ of a PP of intensity
   $$\chi(t) = \lambda(x^{(i-1)} + v^{(i-1)}t, v^{(i-1)}).$$
   (b) Simulate $\tau_{\text{ref}} \sim \text{Exp}\left(\lambda^{\text{ref}}\right)$.
   (c) Set $\tau_i \leftarrow \min\left(\tau_{\text{bounce}}, \tau_{\text{ref}}\right)$ and compute the next position using
   $$x^{(i)} \leftarrow x^{(i-1)} + v^{(i-1)}\tau_i. \tag{3}$$
   (d) If $\tau_i = \tau_{\text{ref}}$, sample the next velocity $v^{(i)} \sim \mathcal{N}\left(0_d, I_d\right)$.
   (e) If $\tau_i = \tau_{\text{bounce}}$, compute the next velocity $v^{(i)}$ using
   $$v^{(i)} \leftarrow R\left(x^{(i)}\right) v^{(i-1)}. \tag{4}$$
   (f) If $t_i = \sum_{j=1}^{i} \tau_j \geq T$ exit For Loop (line 2).

In the algorithm above, $\text{Exp}(\delta)$ denotes the exponential distribution of rate δ and $\mathcal{N}\left(0_d, I_d\right)$ the standard normal on $\mathbb{R}^d$. Refer to Figure 1 for an example of a trajectory generated by BPS on a standard bivariate Gaussian target distribution. An example of a bounce time simulation is shown in Figure 1, top right, for the segment between the first and second events—the intensity $\chi(t)$ (turquoise) is obtained by thresholding $\left\langle \nabla U\left(x^{(1)} + v^{(1)}t\right), v^{(1)} \right\rangle$ (purple, dashed); arrival times of the PP of intensity $\chi(t)$ are denoted by stars.

We will show further that the transition kernel of the BPS process admits π as invariant distribution for any $\lambda^{\text{ref}} \geq 0$ but it can fail to be irreducible when $\lambda^{\text{ref}} = 0$ as demonstrated in Section 4.1. It is thus critical to use $\lambda^{\text{ref}} > 0$. Our proof of invariance and ergodicity can accommodate some alternative refreshment steps 2d. One such variant, which we call restricted refreshment, samples $v^{(i)}$ uniformly on the unit hypersphere $\mathcal{S}^{d-1} = \left\{x \in \mathbb{R}^d : \|x\| = 1\right\}$. We compare experimentally these two variants and others in Section 4.3.

2.3 Algorithms for bounce time simulation

Implementing BPS requires sampling the first arrival time τ of a one-dimensional inhomogeneous PP Π of intensity $\chi(t) = \lambda(x + vt, v)$ given by (1). Simulating such a process is a well-studied problem; see [5, Chapter 6, Section 1.3]. We review here three methods and illustrate how they can be used to implement BPS for examples from Bayesian statistics. The first method described in Section 2.3.1 will be particularly useful when the target is log-concave, while the two others described in Section 2.3.2 and Section 2.3.3 can be applied to more general scenarios.

2.3.1 Simulation using a time-scale transformation

If we let $\Xi(t) = \int_0^t \chi(s)\,\mathrm{d}s$, then the PP Π satisfies

$$\mathbb{P}(\tau > u) = \mathbb{P}(\Pi \cap [0,u) = \emptyset) = \exp(-\Xi(u)),$$

and therefore τ can be simulated from a uniform variate $V \sim \mathcal{U}(0,1)$ via the identity

$$\tau = \Xi^{-1}(-\log(V)), \tag{5}$$

where Ξ^{-1} denotes the quantile function of Ξ, $\Xi^{-1}(p) = \inf\{t : p \leq \Xi(t)\}$. Refer to Figure 1, top right for a graphical illustration. This identity corresponds to the method proposed in [22] to determine the bounce times and is also used in [18, 19, 21] to simulate related processes.

In general, it is not possible to obtain an analytical expression for τ. However, when the target distribution is strictly log-concave and differentiable, it is possible to solve Equation (5) numerically (see Example 1 below).

Example 1. *Log-concave densities.* If the energy is strictly convex (see Figure 1, bottom right), we can minimize it along the line specified by (x, v)

$$\tau_* = \mathrm{argmin}_{t:t\geq 0}\, U(x + vt),$$

where τ_* is well defined and unique by strict convexity. On the interval $[0, \tau_*)$, which might be empty, we have $\mathrm{d}U(x+vt)/\mathrm{d}t < 0$ and $\mathrm{d}U(x+vt)/\mathrm{d}t \geq 0$ on $[\tau_*, \infty)$. The solution τ of (5) is thus necessarily such that $\tau \geq \tau_*$ and (5) can be rewritten using the gradient theorem as

$$\int_{\tau_*}^{\tau} \frac{\mathrm{d}U(x+vt)}{\mathrm{d}t}\mathrm{d}t = U(x+v\tau) - U(x+v\tau_*) = -\log V. \tag{6}$$

Even if we only compute U pointwise through a black box, we can solve (6) through line search within machine precision.

We note that (6) also provides an informal connection between the BPS and MH algorithms. Exponentiating this equation, we get indeed

$$\frac{\pi(x+v\tau)}{\pi(x+v\tau_*)} = V.$$

Hence, in the log-concave case, and when the particle is climbing the energy ladder (i.e., $\tau_* = 0$), BPS can be viewed as "swapping" the order of the steps taken by the MH algorithm. In the latter, we first sample a proposal and second sample a uniform V to perform an accept-reject decision. With BPS, V is first drawn then the maximum distance allowed by the same MH ratio is travelled. As for the case of a particle going down the energy ladder, the behavior of BPS is simpler to understand: bouncing never occurs. We illustrate this method for Gaussian distributions.

Multivariate Gaussian distributions. Let $U(x) = \|x\|^2$, then simple calculations yield

$$\tau = \frac{1}{\|v\|^2}\begin{cases} -\langle x, v\rangle + \sqrt{-\|v\|^2 \log V} & \text{if } \langle x, v\rangle \leq 0, \\ -\langle x, v\rangle + \sqrt{\langle x, v\rangle^2 - \|v\|^2 \log V} & \text{otherwise.} \end{cases} \tag{7}$$

2.3.2 Simulation using adaptive thinning

When it is difficult to solve (5), the use of an adaptive thinning procedure provides an alternative. Assume we have access to local-in-time upper bounds $\bar{\chi}_s(t)$ on $\chi(t)$, that is

$$\bar{\chi}_s(t) = 0 \text{ for all } t < s,$$
$$\bar{\chi}_s(t) \geq \chi(t) \text{ for all } s \leq t \leq s + \Delta(s),$$

where $\triangle$ is a positive function (standard thinning corresponds to $\Delta = +\infty$). Assume additionally that we can simulate the first arrival time of the PP $\bar{\Pi}_s$ with intensity $\bar{\chi}_s(t)$. Such bounds can be constructed based on upper bounds on directional derivatives of U provided the remainder of the Taylor expansion can be controlled. Algorithm 2 shows the pseudocode for the adaptive thinning procedure.

Algorithm 2 Simulation of the first arrival time of a PP through thinning

1. Set $s \leftarrow 0$, $\tau \leftarrow 0$.
2. Do
   (a) Set $s \leftarrow \tau$.
   (b) Sample τ as the first arrival point of the PP $\bar{\Pi}_s$ of intensity $\bar{\chi}_s$.
   (c) If $\bar{\Pi}_s = \{\emptyset\}$ then set $\tau \leftarrow s + \triangle(s)$.
   (d) If $s + \triangle(s) \leq \tau$ set $s \leftarrow s + \triangle(s)$ and go to (b).
   (e) While $V > \{\chi(\tau)/\bar{\chi}_s(\tau)\}$ where $V \sim \mathcal{U}(0,1)$.
3. Return τ.

The case $V > \{\chi(\tau)/\bar{\chi}_s(\tau)\}$ corresponds to a rejection step in the thinning algorithm but, in contrast to rejection steps that occur in standard MCMC samplers, in the BPS algorithm this means that the particle does not bounce and just coasts. Practically, we would like ideally $\triangle$ and the ratio $\chi(\tau)/\bar{\chi}_s(\tau)$ to be large. Indeed this would avoid having to simulate too many candidate events from $\bar{\Pi}_s$ which would be rejected as these rejection steps incur a computational cost.

2.3.3 Simulation using superposition and thinning

Assume that the energy can be decomposed as

$$U(x) = \sum_{j=1}^{m} U^{[j]}(x), \tag{8}$$

then

$$\chi(t) \leq \sum_{j=1}^{m} \chi^{[j]}(t),$$

where $\chi^{[j]}(t) = \max\left(0, \left\langle \nabla U^{[j]}(x+tv), v \right\rangle\right)$ for $j = 1, ..., m$. It is therefore possible to use the thinning algorithm of Section 2.3.2 with $\bar{\chi}_0(t) = \sum_{j=1}^{m} \chi^{[j]}(t)$ for $t \geq 0$ (and $\Delta = +\infty$), as we can simulate from $\bar{\Pi}_0$ via superposition by simulating the first arrival time $\tau^{[j]}$ of each PP with intensity $\chi^{[j]}(t) \geq 0$ then returning

$$\tau = \min_{j=1,...,m} \tau^{[j]}.$$

Example 2. *Exponential families.* Consider a univariate exponential family with parameter x, observation y, sufficient statistic $\phi(y)$ and log-normalizing constant $A(x)$. If we assume a Gaussian prior on x, we obtain

$$U(x) = \underbrace{x^2/2}_{U^{[1]}(x)} + \underbrace{-x\phi(y)}_{U^{[2]}(x)} + \underbrace{A(x)}_{U^{[3]}(x)}.$$

The time $\tau^{[1]}$ is computed analytically in Example 1 whereas the times $\tau^{[2]}$ and $\tau^{[3]}$ are given by

$$\tau^{[2]} = \begin{cases} \frac{\log V^{[2]}}{v\phi(y)} & \text{if } v\phi(y) < 0, \\ +\infty & \text{otherwise,} \end{cases}$$

and

$$\tau^{[3]} = \begin{cases} \tilde{\tau}^{[3]} & \text{if } \tilde{\tau}^{[3]} > 0, \\ +\infty & \text{otherwise,} \end{cases}$$

with $\tilde{\tau}^{[3]} = (A^{-1}(-\log V^{[3]} + A(x)) - x)/v$ and $V^{[2]}, V^{[3]} \sim \mathcal{U}(0,1)$. For example, with a Poisson distribution with natural parameter x, we obtain

$$\tilde{\tau}^{[3]} = \frac{\log(-\log V^{[3]} + \exp(x)) - x}{v}.$$

Example 3. *Logistic regression.* The class label of the data point $r \in \{1, 2, \ldots, R\}$ is denoted by $y_r \in \{0, 1\}$ and its covariate $k \in \{1, 2, \ldots, d\}$ by $\iota_{r,k}$ where we assume that $\iota_{r,k} \geq 0$ (this assumption can be easily relaxed as demonstrated but would make the notation more complicated; see [**?**] for details). The parameter $x \in \mathbb{R}^d$ is assigned a standard Gaussian prior density denoted by ψ, yielding the posterior density

$$\pi(x) \propto \psi(x) \prod_{r=1}^{R} \frac{\exp(y_r \langle \iota_r, x \rangle)}{1 + \exp \langle \iota_r, x \rangle}. \tag{9}$$

Using the superposition and thinning method (Section 2.3.3), simulation of the bounce times can be broken into subproblems corresponding to $R+1$ factors: one factor coming from the prior, with corresponding energy

$$U^{[R+1]}(x) = -\log \psi(x) = \|x\|^2/2 + \text{constant}, \tag{10}$$

and R factors coming from the likelihood of each datapoint, with corresponding energy

$$U^{[r]}(x) = \log(1 + \exp \langle \iota_r, x \rangle) - y_r \langle \iota_r, x \rangle . \tag{11}$$

Simulation of $\tau^{[R+1]}$ is covered in Example 1. Simulation of $\tau^{[r]}$ for $r \in \{1, 2, \ldots, R\}$ can be approached using thinning. In Appendix **??**, we show that

$$\chi^{[r]}(t) \leq \bar{\chi}^{[r]} = \sum_{k=1}^{d} \mathbf{1}[v_k(-1)^{y_r} \geq 0] \iota_{r,k} |v_k|. \tag{12}$$

Since the bound is constant for a given v, we sample $\tau^{[r]}$ by simulating an exponential random variable.

2.4 Estimating expectations

Given a realization of $x(t)$ over the interval $[0, T]$, where T is the total trajectory length, the expectation $\int_{\mathbb{R}^d} \varphi(x) \pi(\mathrm{d}x)$ of a function $\varphi : \mathbb{R}^d \to \mathbb{R}$ with respect to π can be estimated using

$$\frac{1}{T} \int_0^T \varphi(x(t)) \, \mathrm{d}t = \frac{1}{T} \left(\sum_{i=1}^{n-1} \int_0^{\tau_i} \varphi\left(x^{(i-1)} + v^{(i-1)} s \right) \mathrm{d}s + \int_0^{t_n - T} \varphi\left(x^{(n-1)} + v^{(n-1)} s \right) \mathrm{d}s \right);$$

see, e.g., [**?**]. When $\varphi(x) = x_k$, $k \in \{1, 2, \ldots, d\}$, we have

$$\int_0^{\tau_i} \varphi\left(x^{(i-1)} + v^{(i-1)} s \right) \mathrm{d}s = x_k^{(i-1)} \tau_i + v_k^{(i-1)} \frac{\tau_i^2}{2}.$$

When the above integral is intractable, we may just discretize $x(t)$ at regular time intervals to obtain an estimator

$$\frac{1}{L} \sum_{l=0}^{L-1} \varphi(x(l\delta)),$$

where $\delta > 0$ is the mesh size and $L = 1 + \lfloor T/\delta \rfloor$. Alternatively, we could approximate these univariate integrals through quadrature.

2.5 Theoretical results

An informal proof establishing that the BPS with $\lambda^{\text{ref}} = 0$ admits π as invariant distribution is given in [22]. As the BPS process $z(t) = (x(t), v(t))$ is a piecewise deterministic Markov process, the expression of its infinitesimal generator can be established rigourously using [**?**]. We show here that this generator has invariant distribution π whenever $\lambda^{\text{ref}} \geq 0$ and prove that the resulting process is additionally ergodic when $\lambda^{\text{ref}} > 0$. We denote by $\mathbb{E}_z[h(z(t))]$ the expectation of $h(z(t))$ under the law of the BPS process initialized at $z(0) = z$.

Proposition 1. *For any $\lambda^{\text{ref}} \geq 0$, the infinitesimal generator $\mathcal{L}$ of the BPS is defined for any sufficiently regular bounded function $h : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ by*

$$\begin{aligned} \mathcal{L}h(z) &= \lim_{t \downarrow 0} \frac{\mathbb{E}_z[h(z(t))] - h(z)}{t} \\ &= \langle \nabla_x h(x,v), v \rangle + \lambda(x,v)\{h(x, R(x)v) - h(z)\} \\ &\quad + \lambda^{\text{ref}} \int (h(x,v') - h(x,v))\psi(v')\,\mathrm{d}v', \end{aligned} \tag{13}$$

where we recall that $\psi(v)$ denotes the standard multivariate Gaussian density on $\mathbb{R}^d$.

This transition kernel of the BPS is non-reversible and admits ρ as invariant probability measure, where the density of ρ w.r.t. Lebesgue measure on $\mathbb{R}^d \times \mathbb{R}^d$ is given by

$$\rho(z) = \pi(x)\psi(v). \tag{14}$$

If we add the condition $\lambda^{\text{ref}} > 0$, we get the following stronger result.

Theorem 1. *If $\lambda^{\text{ref}} > 0$ then ρ is the unique invariant probability measure of the transition kernel of the BPS and for ρ-almost every $z(0)$ and h integrable with respect to ρ*

$$\lim_{T \to \infty} \frac{1}{T} \int_0^T h(z(t))\mathrm{d}t = \int h(z)\rho(z)\mathrm{d}z \qquad a.s.$$

We exhibit in Section 4.1 a simple example where P_t is not ergodic for $\lambda^{\text{ref}} = 0$.

3 The local bouncy particle sampler

3.1 Structured target distribution and factor graph representation

In numerous applications, the target distribution admits some structural properties that can be exploited by sampling algorithms. For example, the Gibbs sampler takes advantage of conditional independence properties. We present here a "local" version of the BPS introduced in [22, Section III] which can similarly exploit these properties and, more generally, any representation of the target density as a product of positive factors

$$\pi(x) \propto \prod_{f \in F} \gamma_f(x_f), \tag{15}$$

where x_f is a restriction of x to a subset $N_f \subseteq \{1,2,\ldots,d\}$ of the components of x, and F is an index set called the set of factors. Hence the energy associated to π is of the form

$$U(x) = \sum_{f \in F} U_f(x) \tag{16}$$

with $\partial U_f(x)/\partial x_k = 0$ for any variable absent from factor f, i.e. for any $k \in \{1,2,\ldots,d\} \setminus N_f$.

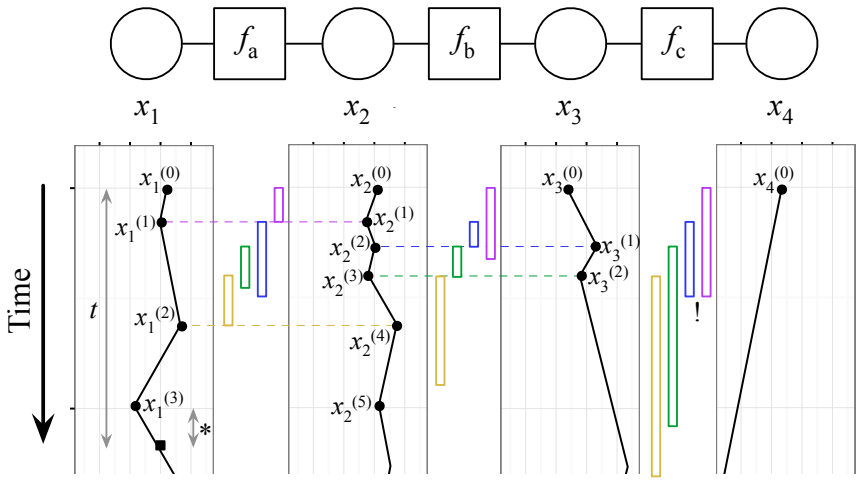

Figure 2: Top: a factor graph with $d = 4$ variables and 3 binary factors, $F = \{f_a, f_b, f_c\}$. Bottom: sample paths of $(x_i(t))_{t\geq 0}$ for $i = 1, ..., 4$ for the local BPS. See Sections 3.2 and 3.3.1.

Such a factorization of the target density can be formalized using factor graphs (Figure 2, top). A factor graph is a bipartite graph, with one set of vertices N called the variables, each corresponding to a component of x ($|N| = d$), and a set of vertices F corresponding to the local factors $(\gamma_f)_{f\in F}$. There is an edge between $k \in N$ and $f \in F$ if and only if $k \in N_f$. This representation generalizes undirected graphical models [27, Chap. 2, Section 2.1.3] as, for example, factor graphs can have distinct factors connected to the same set of components (i.e. $f \neq f'$ with $N_f = N_{f'}$) as in the example of Section 4.6.

3.2 Local BPS: algorithm description

Similarly to the Gibbs sampler, each step of the local BPS manipulates only a subset of the d components of x. Contrary to the Gibbs sampler, the local BPS does not require sampling from any full conditional distribution and each local calculation results in a change of *all* state components, not just the one being updated—how this can be done implicitly without manipulating the full state at each iteration is described below. Related processes exhibiting similar characteristics have been proposed in [15, 18, 19, 21].

For each factor $f \in F$, we define a local intensity function $\lambda_f : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^+$ and a local bouncing matrix $R_f : \mathbb{R}^d \to \mathbb{R}^{d\times d}$ by

$$\lambda_f(x, v) = \max\{0, \langle \nabla U_f(x), v\rangle\}, \tag{17}$$

$$R_f(x)\, v = v - 2\frac{\langle \nabla U_f(x), v\rangle \nabla U_f(x)}{\|\nabla U_f(x)\|^2}. \tag{18}$$

We can check that $R_f(x)$ satisfies

$$k \in \{1, 2, \ldots, d\} \setminus N_f \Longrightarrow \{R_f(x)v\}_k = v_k. \tag{19}$$

When suitable, we will slightly abuse notation and write $R_f(x_f)$ for $R_f(x)$ as $R_f(x_f, x_{-f}) = R_f\left(x_f, x'_{-f}\right)$ for any $x_{-f}, x'_{-f} \in \mathbb{R}^{d-|N_f|}$, where $|S|$ denotes the cardinality of a set S. Similarly, we will use $\lambda_f(x_f, v_f)$ for $\lambda_f(x, v)$.

We define a collection of PP intensities based on the previous event position $x^{(i-1)}$ and velocity $v^{(i-1)}$: $\chi_f(t) = \lambda_f(x^{(i-1)} + v^{(i-1)}t, v^{(i-1)})$. In the local BPS, the next bounce time τ is the first

arrival of a PP with intensity $\chi(t) = \sum_{f \in F} \chi_f(t)$. However, instead of modifying all velocity variables at a bounce as in the basic BPS, we sample a factor f with probability $\chi_f(\tau)/\chi(\tau)$ and modify only the variables connected to the sampled factor. More precisely, the velocity v_f is updated using $R_f(x_f)$ defined in (18). A generalization of the proof of Proposition 1 given in the Supplementary Material shows that the local BPS algorithm results in a $\pi-$invariant kernel. In the next subsection, we describe various computationally efficient procedures to simulate this process.

For all these implementations, it is useful to encode trajectories in a sparse fashion: each variable $k \in N$ only records information at the times $t_k^{(1)}, t_k^{(2)}, \ldots$ where an event (a bounce or refreshment) affected it. By (19), this represents a sublist of the list of all event times. At each of those times $t_k^{(i)}$, the component's position $x_k^{(i)}$ and velocity $v_k^{(i)}$ right after the event is stored. Let L_k denote a list of triplets $(x_k^{(i)}, v_k^{(i)}, t_k^{(i)})_{i \geq 0}$, where $x_k^{(0)}$ and $v_k^{(0)}$ denote the initial position and velocity and $t_k^{(0)} = 0$ (see Figure 2, where the black dots denote the set of recorded triplets). This list is sufficient to compute $x_k(t)$ for $t \leq t_k^{(|L_k|+1)}$. This procedure is detailed in Algorithm 3 and an example is shown in Figure 2, where the black square on the first variable's trajectory shows how Algorithm 3 reconstructs $x_1(t)$ at a fixed time t: it identifies $i(t,1) = 3$ as the index associated to the largest event time $t_1^{(3)}$ before time t affecting x_1 and return $x_1(t) = x_1^{(3)} + v_1^{(3)}(t - t_1^{(3)})$.

Algorithm 3 Computation of $x_k(t)$ from a list of events.

1. Find the index $i = i(t,k)$ associated to the largest time $t_k^{(i)}$ verifying $t_k^{(i)} \leq t$.
2. Set $x_k(t) \leftarrow x_k^{(i(t,k))} + (t - t_k^{(i(t,k))}) v_k^{(i(t,k))}$.

3.3 Local BPS: efficient implementations

3.3.1 Implementation via priority queue

We can sample arrivals from a PP with intensity $\chi(t) = \sum_{f \in F} \chi_f(t)$ using the superposition method of Section 2.3.3, the thinning step therein being omitted. To implement this technique efficiently, we store potential future bounce times (called "candidates") t_f, one for each factor, in a priority queue Q: only a subset of these candidates will join the lists L_k which store past, "confirmed" events. We pick the the smallest time in Q to determine the next bounce time and the next factor f to modify. The priority queue structure ensures that finding the minimum element of Q or inserting/updating an element of Q can be performed with computational complexity $O(\log |F|)$. When a bounce occurs, a key observation behind efficient implementation of the local BPS is that not all the other candidate bounce times need to be resimulated. Suppose that the bounce was associated with factor f. In this case, only the candidate bounce times $t_{f'}$ corresponding to factors f' with $N_{f'} \cap N_f \neq \emptyset$ need to be resimulated. For example, consider the first bounce in Figure 2 (shown in purple), which is triggered by factor f_a (rectangles represent candidate bounce times t_f; dashed lines connect bouncing factors to the variables that undergo an associated velocity change). Then only the velocities for the variables x_1 and x_2 need to be updated. Therefore, only the candidate bounce times for factors f_a and f_b need to be re-simulated while the candidate bounce time for f_c stays constant (this is shown by an exclamation mark in Figure 2).

The method is detailed in Algorithm 4. Several operations of the BPS such as step 4, 6.iii, 6.iv and 7.ii can be easily parallelized.

Algorithm 4 Local BPS algorithm (priority queue implementation)

1. Initialize $\left(x^{(0)}, v^{(0)}\right)$ arbitrarily on $\mathbb{R}^d \times \mathbb{R}^d$.
2. Initialize the global clock $T \leftarrow 0$.
3. For $k \in N$ do
   (a) Initialize the list $L_k \leftarrow \left(x_k^{(0)}, v_k^{(0)}, T\right)$.
4. Set $Q \leftarrow$ **new queue** $\left(x^{(0)}, v^{(0)}, T\right)$.
5. Sample $t_{\text{ref}} \sim \text{Exp}\left(\lambda^{\text{ref}}\right)$.
6. While more events $i = 1, 2, \ldots$ requested do
   (a) $(t, f) \leftarrow$ **smallest candidate bounce time and associated factor in** Q.
   (b) Remove (t, f) from Q.
   (c) Update the global clock, $T \leftarrow t$.
   (d) If $T < t_{\text{ref}}$ then
      i. $(v_f)_k \leftarrow v_k^{(|L_k|-1)}$ for all $k \in N_f$.
      ii. $x_f \leftarrow x_f(T)$ (computed using Algorithm 3).
      iii. For $k \in N_f$ do
         A. $x_k^{(|L_k|)} \leftarrow x_k^{(|L_k|-1)} + (T - t_k^{(|L_k|-1)}) v_k^{(|L_k|-1)}$, where $t_k^{(|L_k|-1)}$ and $v_k^{(|L_k|-1)}$ are retrieved from L_k.
         B. $v_k^{(|L_k|)} \leftarrow \{R_f(x_f) v_f\}_k$.
         C. $L_k \leftarrow \left\{L_k, \left(x_k^{(|L_k|)}, v_k^{(|L_k|)}, T\right)\right\}$ (add the new sample to the list).
      iv. For $f' \in F : N_{f'} \cap N_f \neq \emptyset$ (note: this includes the update of f) do
         A. for all $k \in N_{f'}$.
         B. $x_{f'} \leftarrow x_{f'}(T)$ (computed using Algorithm 3).
         C. Simulate the first arrival time $\tau_{f'}$ of a PP of intensity $\lambda_{f'}(x_{f'} + t v_{f'}, v_{f'})$ on $[0, +\infty)$.
         D. Set in Q the candidate bounce time associated to f' to the value $t_{f'} = T + \tau_{f'}$.
   (e) Else
      i. Sample $v' \sim \mathcal{N}(0_d, I_d)$.
      ii. $Q \leftarrow$ **new queue** $(x(t_{\text{ref}}), v', t_{\text{ref}})$ where $x(t_{\text{ref}})$ is computed using Algorithm 3.
      iii. Set $t_{\text{ref}} \leftarrow t_{\text{ref}} + \tau_{\text{ref}}$ where $\tau_{\text{ref}} \sim \text{Exp}\left(\lambda^{\text{ref}}\right)$.
7. Return the samples encoded as the lists L_k, $k \in N$.

Algorithm 5 New Queue (x, v, T)

1. For $f \in F$ do
   (a) $(v_f)_k \leftarrow v_k^{(|L_k|-1)}$ for all $k \in N_f$.
   (b) $x_f \leftarrow x_f(T)$ (computed using Algorithm 3).
   (c) Simulate the first arrival time τ_f of a PP of intensity $\lambda_f(x_f + t v_f, v_f)$ on $[0, +\infty)$.
   (d) Set in Q the time associated to f to the value $T + \tau_f$.
2. Return Q.

3.3.2 Implementation via thinning

When the number of factors involved in Step 6(d)iv is large, the previous queue-based implementation can be computationally expensive. Implementing the local BPS in this setup is closely related to the problem of simulating stochastic chemical kinetics and innovative solutions have been proposed in this area. We adapt here the algorithm proposed in [26] to the local BPS context. For ease of presentation, we present the algorithm without refreshment and only detail the simulation of the bounce times. This algorithm relies on the ability to compute local-in-time upper bounds on λ_f for all $f \in F$. More precisely, we assume that given a current position x and velocity v, and $\Delta \in (0, \infty]$, we can find a positive number $\bar{\chi}_f$, such that for any $t \in [0, \Delta)$, we have $\bar{\chi}_f \geq \lambda_f(x + vt, v)$. We can also use this method on a subset G of F and combine it with the previously discussed techniques to sample candidate bounce times for factors in $F \backslash G$ but we restrict ourselves to $G = F$ to simplify the presentation.

Algorithm 6 Local BPS algorithm (thinning implementation)

1. Initialize $\left(x^{(0)}, v^{(0)}\right)$ arbitrarily on $\mathbb{R}^d \times \mathbb{R}^d$.
2. Initialize the global clock $T \leftarrow 0$.
3. Initialize $\bar{T} \leftarrow \triangle$ (time until which local upper bounds are valid).
4. Compute local-in-time upper bounds $\bar{\chi}_f$ for $f \in F$ such that $\bar{\chi}_f \geq \lambda_f(x^{(0)} + v^{(0)}t, v^{(0)})$ for all $t \in [0, \Delta)$.
5. While more events $i = 1, 2, \ldots$ requested do
   (a) Sample $\tau \sim \text{Exp}\left(\bar{\chi}\right)$ where $\bar{\chi} = \sum_{f \in F} \bar{\chi}_f$.
   (b) If $\left(T + \tau > \bar{T}\right)$ then
      i. $x^{(i)} \leftarrow x^{(i-1)} + v^{(i-1)}(\bar{T} - T)$.
      ii. $v^{(i)} \leftarrow v^{(i-1)}$.
      iii. For all $f \in F$, update $\bar{\chi}_f$ to ensure that $\bar{\chi}_f \geq \lambda_f(x^{(i)} + v^{(i)}t, v^{(i)})$ for $t \in [0, \Delta)$.
      iv. Set $T \leftarrow \bar{T}$, $\bar{T} \leftarrow \bar{T} + \triangle$.
   (c) Else
      i. $x^{(i)} \leftarrow x^{(i-1)} + v^{(i-1)}\tau$.
      ii. Sample $\mathcal{F} \in F$ where $\mathbb{P}\left(\mathcal{F} = f\right) = \bar{\chi}_f / \bar{\chi}$.
      iii. If $V < \lambda_{\mathcal{F}}\left(x^{(i)}, v^{(i-1)}\right) / \bar{\chi}_{\mathcal{F}}$ where $V \sim \mathcal{U}\left(0, 1\right)$ then a bounce for factor $\mathcal{F}$ occurs at time T.
         A. $v^{(i)} \leftarrow R_{\mathcal{F}}\left(x^{(i)}\right) v^{(i-1)}$.
         B. For all $f' \in F : N_{f'} \cap N_{\mathcal{F}} \neq \emptyset$, update $\bar{\chi}_{f'}$ to ensure that $\bar{\chi}_{f'} \geq \lambda_{f'}(x^{(i)} + v^{(i)}t, v^{(i)})$ for $t \in [0, \bar{T} - T - \tau)$.
      iv. Else
         A. $v^{(i)} \leftarrow v^{(i-1)}$.
      v. Set $T \leftarrow T + \tau$.

Algorithm 6 will be particularly useful in scenarios where summing over the bounds (Step 5a) and sampling a factor (Step 5(c)ii) can be performed efficiently. A scenario where it is possible to implement these two operations in constant time is detailed in Section 4.6. Another scenario where sampling quickly from $\mathcal{F}$ is feasible is if the number of distinct upper bounds is much smaller than the number of factors. For example, we only need to sample a factor $\mathcal{F}$ uniformly at random if $\Lambda = \bar{\chi}_f = \bar{\chi}_{f'}$ for all f, f' in F and $\bar{\chi} = |F| \cdot \Lambda$ (that is no factor needs to be inspected in order to execute Algorithm 5(c)ii) and the thinning procedure in Step 5(c)iii boils down to

$$V \leq \frac{|F| \max\left(0, \left\langle \nabla U_f(x^{(i)}), v^{(i-1)} \right\rangle\right)}{\bar{\chi}}. \tag{20}$$

A related approach has been adopted in [**?**] for the analysis of big data. In this particular scenario, an alternative local BPS can also be implemented where $s > 1$ factors $\mathcal{F} = (\mathcal{F}_1, \ldots, \mathcal{F}_s)$ are sampled uniformly at random without replacement from F, the thinning occurs with probability

$$\frac{|F|}{s\bar{\chi}} \max\left(0, \sum_{j=1}^{s} \left\langle \nabla U_{\mathcal{F}_j}(x^{(i)}), v^{(i-1)} \right\rangle\right) \tag{21}$$

and the components of x belonging to $N_{\mathcal{F}}$ bounce based on $\sum_{j=1}^{s} \nabla U_{\mathcal{F}_j}(x)$. One can check that the resulting dynamics preserves π as an invariant distribution. In contrast to $s = 1$, this is not an implementation of local BPS described in Algorithm 6, but instead this corresponds to a local BPS update for a random partition of the factors.

4 Numerical results

4.1 Gaussian distributions and the need for refreshment

We consider an isotropic multivariate Gaussian target distribution, $U(x) = \|x\|^2$, to illustrate the need for refreshment. Without refreshment, we obtain from Equation (7)

$$\left\langle x^{(i)}, v^{(i)} \right\rangle = \begin{cases} -\sqrt{-\log V_i} & \text{if } \left\langle x^{(i-1)}, v^{(i-1)} \right\rangle \leq 0, \\ -\sqrt{\left\langle x^{(i-1)}, v^{(i-1)} \right\rangle^2 - \log V_i} & \text{otherwise,} \end{cases}$$

and

$$\left\|x^{(i)}\right\|^2 = \begin{cases} \left\|x^{(i-1)}\right\|^2 - \left\langle x^{(i-1)}, v^{(i-1)} \right\rangle^2 - \log V_i & \text{if } \left\langle x^{(i-1)}, v^{(i-1)} \right\rangle \leq 0, \\ \left\|x^{(i-1)}\right\|^2 - \log V_i & \text{otherwise,} \end{cases}$$

see Supplementary Material for details. In particular, these calculations show that if $\left\langle x^{(i)}, v^{(i)} \right\rangle \leq 0$ then $\left\langle x^{(j)}, v^{(j)} \right\rangle \leq 0$ for $j > i$ so that $\|x^{(i)}\|^2 = \left\|x^{(1)}\right\|^2 - \left\langle x^{(1)}, v^{(1)} \right\rangle^2 - \log V_i$ for $i \geq 2$. In particular for $x^{(0)} = e_1$ and $v^{(0)} = e_2$ with e_i being elements of standard basis of $\mathbb{R}^d$, the norm of the position at all points along the trajectory can never be smaller than 1 as illustrated in Figure 3.

In this scenario, we show that BPS without refreshment admits a countably infinite collection of invariant distributions. Let us define $r(t) = \|x(t)\|$ and $m(t) = \langle x(t), v(t) \rangle / \|x(t)\|$ and denote by χ_k the probability density of the chi distribution with k degrees of freedom.

Proposition 2. *For any dimension $d \geq 2$, the process $(r(t), m(t))_{t \geq 0}$ is Markov and its transition kernel is invariant with respect to the probability densities $\left\{ f_k(r, m) \propto \chi_k(\sqrt{2}r) \cdot (1 - m^2)^{(k-3)/2}; k \in \{2, 3, \ldots\} \right\}$.*

The proof is given in Appendix 2. By Theorem 1, we have a unique invariant measure as soon as $\lambda^{\text{ref}} > 0$.

Next, we look at the scaling of the Effective Sample Size (ESS) per CPU second of the basic BPS algorithm for $\varphi(x) = x_1$ when $\lambda_{\text{ref}} = 1$ as the dimension d of the isotropic normal target increases. The ESS is estimated using the R package *mcmcse* [7] by evaluating the trajectory on a fine discretization of the sampled trajectory. The results in log-log scale are displayed in Figure 3. The curve suggests a decay of roughly $d^{-1.47}$, slightly inferior to the $d^{-1.25}$ scaling for

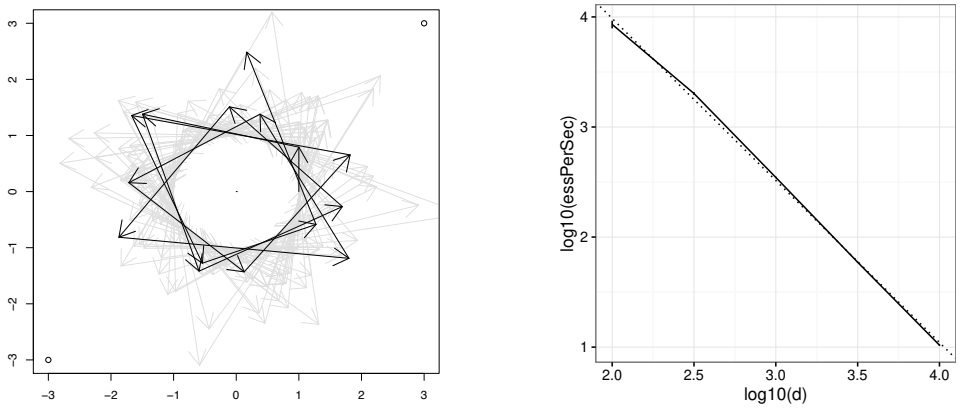

Figure 3: Left: the 200 first segments/bounces of a BPS path for $\lambda^{\text{ref}} = 0$ (for clarity the first 15 segments are in black, the following ones in light grey): the center of the space is never explored. Right, solid line: ESS per CPU second as a function of d (log-log scale), along with 95% confidence intervals based on 40 runs (the intervals are small and difficult to see). Dashed line: linear regression curve. See Section 4.1 for details.

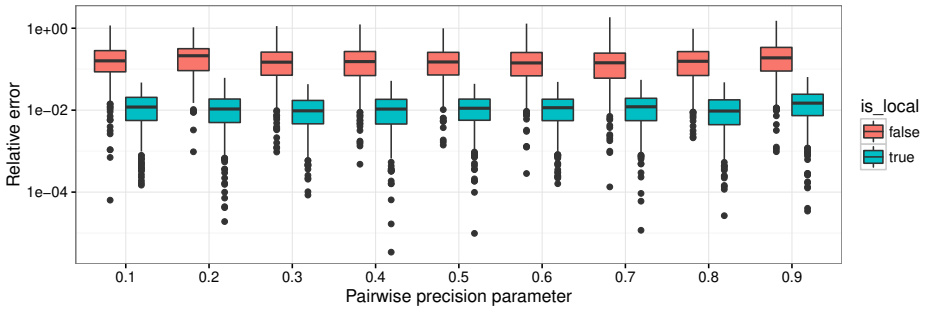

Figure 4: Boxplots of relative errors over 100 local BPS runs for Gaussian chain-shaped fields of pairwise precisions 0.1-0.9 .

an optimally tuned Hamiltonian Monte Carlo (HMC) algorithm [4, Section III], [20, Section 5.4.4]. It should be noted that BPS achieves this scaling without varying any tuning parameter, whereas HMC's performance critically depends on tuning two parameters (leap-frog stepsize and number of leap-frog steps). Both BPS and HMC compare favorably to the d^{-2} scaling of the optimally tuned random walk MH [23].

4.2 Comparison of the global and local schemes

We compare the basic "global" BPS of Section 2 to the local BPS of Section 3 on a sparse Gaussian field. We use a chain-shaped undirected graphical model of length $d = 1000$ and perform separate experiments for various pairwise precision parameters for the interaction between neighbors in the chain. Both methods are run for 60 seconds. We compare the Monte Carlo estimate of the variance of x_{500} to its true value. The results are shown in Figure 4. The smaller computational complexity per local bounce of the local BPS offsets significantly the associated decrease in expected trajectory segment length. Moreover, both versions appear insensitive to the pairwise precision used in this sparse Gaussian field.

4.3 Comparisons of alternative refreshment schemes

In Section 2, the velocity was refreshed using a Gaussian distribution. We compare here this global refreshment scheme to three alternatives:

Local refreshment: if the local BPS is used, the factor graph structure can be exploited to design computationally cheaper refreshment operators. We pick one factor $f \in F$ uniformly at random and resample only the components of v with indices in N_f. By the same argument used in Section 3, each refreshment requires bounce time recomputation only for the factors f' with $N_f \cap N_{f'} \neq \emptyset$.

Restricted refreshment: the velocities are refreshed according to $\phi(v)$, the uniform distribution on $\mathcal{S}^{d-1}$, and the BPS admits now $\rho(z) = \pi(x)\,\phi(v)$ as invariant distribution.

Restricted partial refreshment: a variant of restricted refreshment where we sample an angle θ by multiplying a Beta(α, β)-distributed random variable by 2π. We then select a vector uniformly at random from the unit length vectors that have an angle θ from v. We used $\alpha = 1, \beta = 4$ to favor small angles.

We compare these methods for different values of λ^{ref}, the trade-off being that too small a value can lead to a failure to visit certain regions of the space, while too large a value leads to a random walk behavior.

The rationale behind the partial refreshment procedure is to suppress the random walk behavior of the particle path arising from a refreshment step independent from the current velocity. Refreshment is needed to ensure ergodicity but a "good" direction should only be altered slightly. This strategy is akin to the partial momentum refreshment strategy for HMC methods [12], [20, Section 4.3] and could be similarly implemented for global refreshment. It is easy to check that all of the above schemes preserve π as invariant distribution. We tested these schemes on the chain-shaped factor graph described in the previous section (with the pairwise precision parameter set to 0.5). All methods are provided with a computational budget of 30 seconds. The results are shown in Figure 5. The results show that local refreshment is less sensitive to λ^{ref}, performing as well or better than global refreshment. The performance of the restricted and partial methods appears more sensitive to λ^{ref} and generally inferior to the other two schemes.

One limitation of the results in this section is that the optimal refreshment scheme and refreshment rate will in general be problem dependent. Adaptation methods used in the HMC literature could potentially be adapted to this scenario [28, 11], but we leave these extensions to future work.

4.4 Comparisons with HMC methods on high-dimensional Gaussian distributions

We compare the local BPS with no partial refreshment and $\lambda^{\text{ref}} = 1$ to advanced adaptive versions of HMC implemented using Stan [11] on a 100-dimensional Gaussian example from [20, Section 5.3.3.4]. For each method, we compute the relative error on the estimated marginal variances after a wall clock time of 30 seconds, excluding from this time the time taken to compile the Stan program. The adaptive HMC methods use 1000 iterations of adaptation. When only the leap-frog stepsize is adapted ("adapt=true"), HMC provides several poor estimates of marginal variances. These deviations disappear when adapting a diagonal metric (denoted "fit-metric") and/or using advanced auxiliary variable methods to select the number of leap-frog steps (denoted "nuts"). Given that adaptation is critical to HMC in this scenario, it is encouraging that BPS without adaptation is competitive (Figure 6).

Next, we compare the local BPS to NUTS ("adapt=true,fit_metric=true,nuts=true") as the dimension d increases. Experiments are performed on the chain-shaped Gaussian Random Field of Section 4.2 with the pairwise precision parameter set to 0.5. We vary the length of the chain (10,

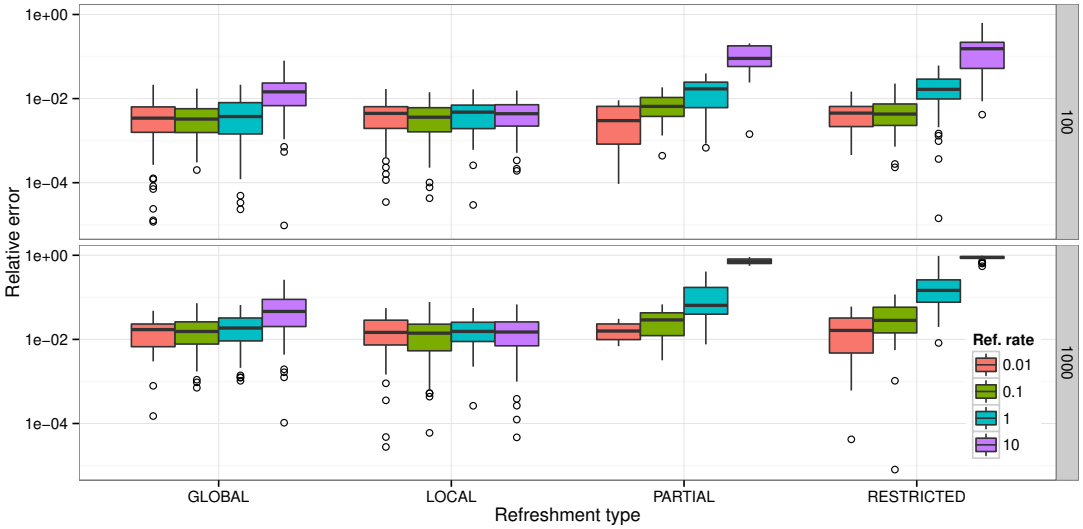

Figure 5: Comparison of refreshment schemes for $d = 100$ (top) and $d = 1000$ (bottom). Each boxplot summarizes the relative error for the variance estimates (in log scale) of x_{50} over 100 runs of BPS.

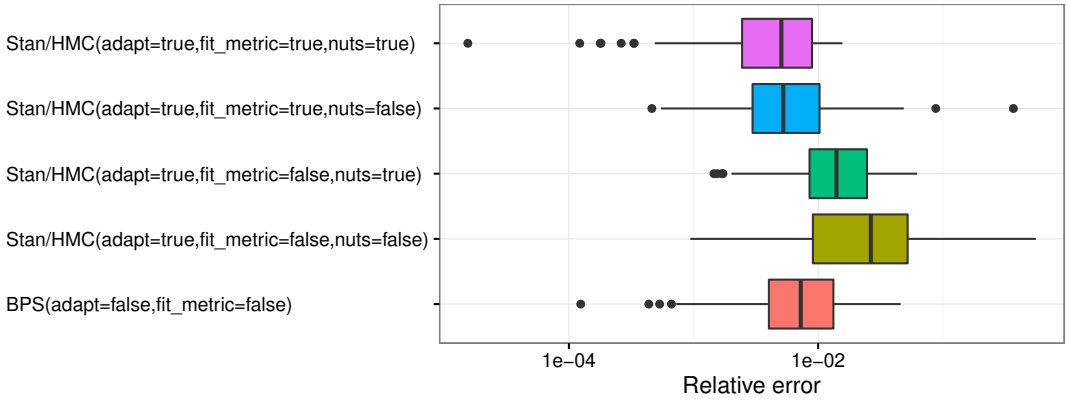

Figure 6: Box plots showing the relative absolute error of variance estimates for a fixed computational budget.

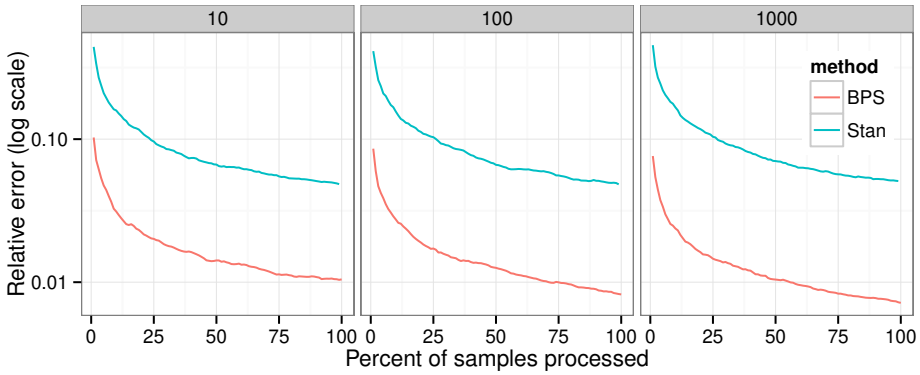

Figure 7: Relative error for $d = 10$ (left), $d = 100$ (middle) and $d = 1000$ (right), averaged over 10 of the dimensions and 40 runs. Each d uses a fixed computational budget.

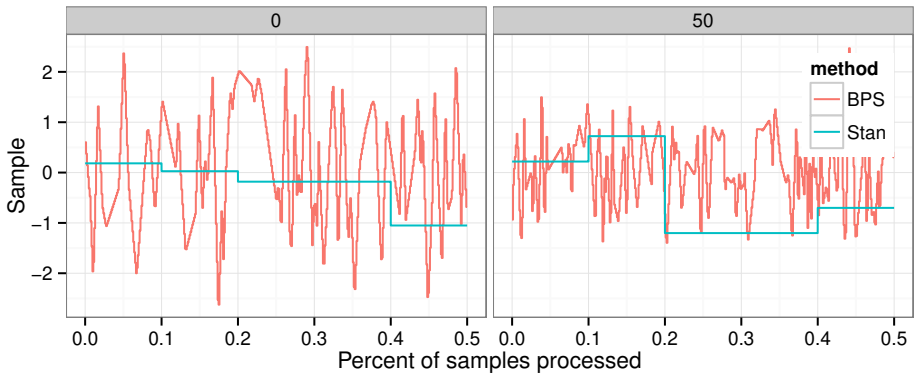

Figure 8: Simulated paths for x_0 and x_{50} for $d = 100$. Each state of the HMC trajectory is obtained by leap-frog steps (not displayed), these latter cannot be used to estimate Monte Carlo averages as HMC relies on a MH step. In contrast, BPS exploits the full path.

100, 1000), and run Stan's implementation of NUTS for 1000 iterations + 1000 iterations of adaptation. We measure the wall-clock time (excluding the time taken to compile the Stan program) and then run our method for the same wall-clock time 40 times for each chain size. The absolute value of the relative error averaged on 10 equally spaced marginal variances is measured as a function of the percentage of the total computational budget used; see Figure 7. The gap between the two methods widens as d increases. To visualize the different behavior of the two algorithms, three marginals of the Stan and BPS paths for $d = 100$ are shown in Figure 8. Contrary to Section 4.1, BPS outperforms here HMC as its local version is able to exploit the sparsity of the random field.

4.5 Poisson-Gaussian Markov random field

Let $x = (x_{i,j} : i, j \in \{1, 2, \dots, 10\})$ denote a grid-shaped Gaussian Markov random field with pairwise interactions of the same form as those used in the previous chain examples (pairwise precision set to 0.5) and let $y_{i,j}$ be Poisson distributed, independent over i, j given x, with rate $\exp(x_{i,j})$. We generate a synthetic dataset $y = (y_{i,j} : i, j \in \{1, 2, \dots, 10\})$ from this model and approximate the resulting posterior distribution of x. We first run Stan with default settings ("adapt=true,fit_metric=true,nuts=true") for $16, 32, 64, \dots, 4096$ iterations. For each number of Stan iterations, we run local BPS for the same wall-clock time as Stan, using a local refreshment ($\lambda^{\text{ref}} = 1$) and the method from Example 2 for the bouncing time computations. We repeat these experiments 10 times with different random seeds. Figure 9 displays the boxplots of the

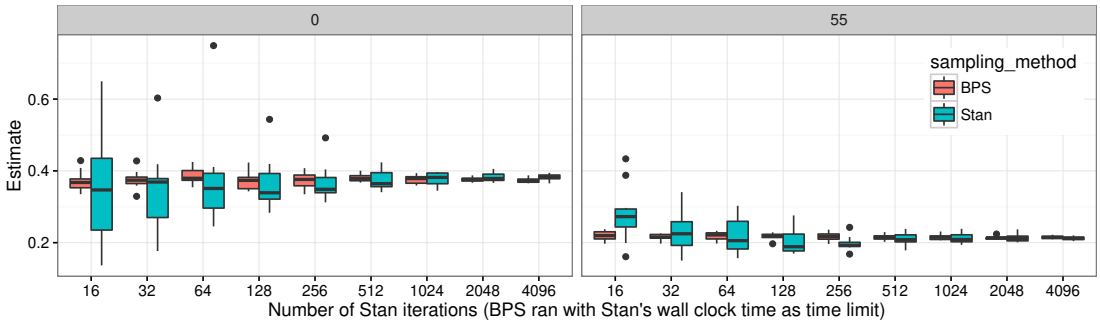

Figure 9: Boxplots of estimates of the posterior variance of $x_{0,0}$ (left) and $x_{5,5}$ (right) using Stan implementation of HMC and local BPS.

estimates of the posterior variances of the variables $x_{0,0}$ and $x_{5,5}$ summarizing the 10 replications. As expected, both methods converge to the same value, but BPS requires markedly less computing time to achieve any given level of accuracy.

4.6 Bayesian logistic regression for large data sets

Consider the logistic regression model introduced in Example 3 when the number of data R is large. In this context, standard MCMC schemes such as the MH algorithm are computationally expensive as they require evaluating the likelihood associated to the R observations at each iteration. This has motivated the development of techniques which only evaluate the likelihood of a subset of the data at each iteration. However, most of the methods currently available introduce either some non-vanishing asymptotic bias, e.g. the subsampling MH scheme proposed in [2], or provide consistent estimates converging at a slower rate than regular MCMC algorithms, e.g. the Stochastic Gradient Langevin Dynamics introduced in [29, 25]. The only available algorithm which only requires evaluating the likelihood of a subset of data at each iteration yet provides consistent estimates converging at the standard Monte Carlo rate is the Firefly algorithm [17].

In this context, we associate $R+1$ factors to the target posterior distribution: one for the prior and one for each data point with $x_f = x$ for all $f \in F$. As a uniform upper bound on the intensities of these local factors is available for restricted refreshment, see Appendix **??**, we could use (21) in conjunction with Algorithm 6 to provide an alternative to the Firefly algorithm which selects at each bounce a subset of s data points uniformly at random without replacement. For $s = 1$, a related algorithm has been recently explored in [**?**]. In presence of outliers, this strategy can be inefficient as the uniform upper bound becomes very large, resulting in a computationally expensive implementation. After a pre-computation step of complexity $O(R \log R)$ only executed once, we show here that Algorithm 6 can be implemented using data-dependent bounds mitigating the sensitivity to outliers while maintaining the computational cost of each bounce independent of R. We first pre-compute the sum of covariates over the data points, $\iota_k^c = \sum_{r=1}^R \iota_{r,k} \mathbf{1}[y_r = c]$, for $k \in \{1, \dots, d\}$ and class label $c \in \{0, 1\}$. Using these quantities, it is possible to compute

$$\bar{\chi} = \sum_{r=1}^{R} \bar{\chi}^{[r]} = \sum_{k=1}^{d} |v_k| \, \iota_k^{\mathbf{1}[v_k < 0]},$$

with $\bar{\chi}^{[r]}$ given in (12). If d is large, we can keep the sum $\bar{\chi}$ in memory and add-and-subtract any local updates to it. The implementation of Step 5(c)ii relies on the alias method [5, Section 3.4]. A detailed description of these derivations and of the algorithm is presented in Appendix **??**.

We compare the local BPS with thinning to the MAP-tuned Firefly algorithm implementation provided by the authors. This version of Firefly outperforms experimentally significantly the standard MH in terms of ESS per-datum likelihood [17]. The two algorithms are here compared in terms of this criterion, where the ESS is averaged over the $d = 5$ components of x. We generate

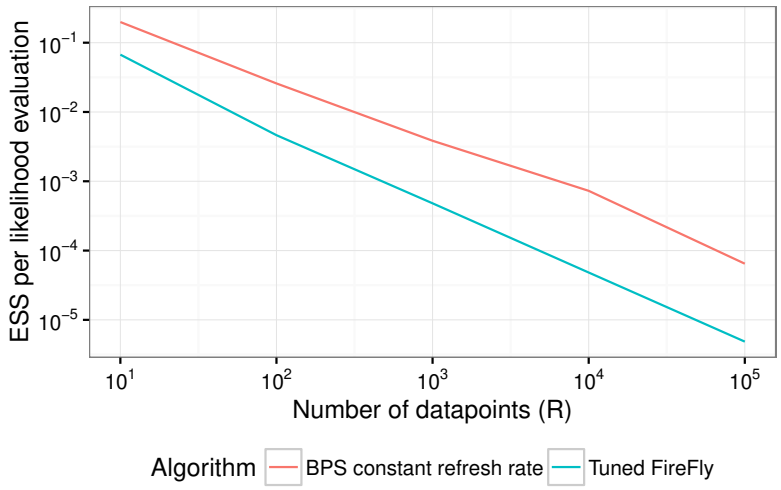

Figure 10: ESS per-datum likelihood evaluation for Local BPS and Firefly.

covariates $\iota_{rk} \overset{\text{i.i.d.}}{\sim} \mathcal{U}(0.1, 1.1)$ and data $y_r \in \{0, 1\}$ for $r = 1, \ldots, R$ according to (9) and set a zero-mean normal prior of covariance $\sigma^2 I_d$ for x. For the algorithm, we set $\lambda_{\text{ref}} = 0.5$, $\sigma^2 = 1$ and $\Delta = 0.5$, which is the length of the time interval for which a constant upper bound for the rate associated with the prior is used, see Algorithm **??**. Experimentally, local BPS always outperforms Firefly, by about an order of magnitude for large data sets. However, we also observe that both Firefly and local BPS have an ESS per datum likelihood evaluation decreasing in approximately $1/R$ so that the gains brought by these algorithms over a correctly scaled random walk MH algorithm do not appear to increase with R. The rate for local BPS is slightly superior in the regime of up to 10^4 data points, but then returns to the approximate $1/R$ rate. To improve this rate, one can adapt the control variate ideas introduced in [**?**] for the MH algorithm to these schemes. This has been proposed in [**?**] for a related algorithm and in [**?**] for the local BPS.

4.7 Bayesian inference of evolutionary parameters

We analyze a dataset of primate mitochondrial DNA [10] at the leaves of a fixed reference tree [13] containing 898 sites and 12 species. We want to approximate the posterior evolutionary parameters x encoded into a rate matrix Q. A detailed description of this phylogenetic model can be found in the Supplementary Material. The global BPS is used with restricted refreshment and $\lambda^{\text{ref}} = 1$ in conjunction with an auxiliary variable-based method similar to the one described in [30], alternating between two moves: (1) sampling continuous-time Markov chain paths along the tree given x using uniformization, (2) sampling x given the path (in which case the derivation of the gradient is simple and efficient). The only difference compared to [30] is that we substitute the HMC kernel by the kernel induced by running BPS for a fixed trajectory length. This auxiliary variable method is convenient because, conditioned on the paths, the energy function is convex and hence we can simulate the bouncing times using the method described in Example 1.

We compare against a state-of-the-art HMC sampler [28] that uses Bayesian optimization to adapt the key parameters of HMC, the leap-frog stepsize and the number of leap-frog steps, while preserving convergence to the target distribution. Both our method and this HMC method are implemented in Java and share the same gradient computation code. Refer to the Supplementary Material for additional background and motivation behind this adaptation method.

We first perform various checks to ensure that both BPS and HMC chains are in close agreement given a sufficiently large number of iterations. After 20 millions HMC iterations, the credible

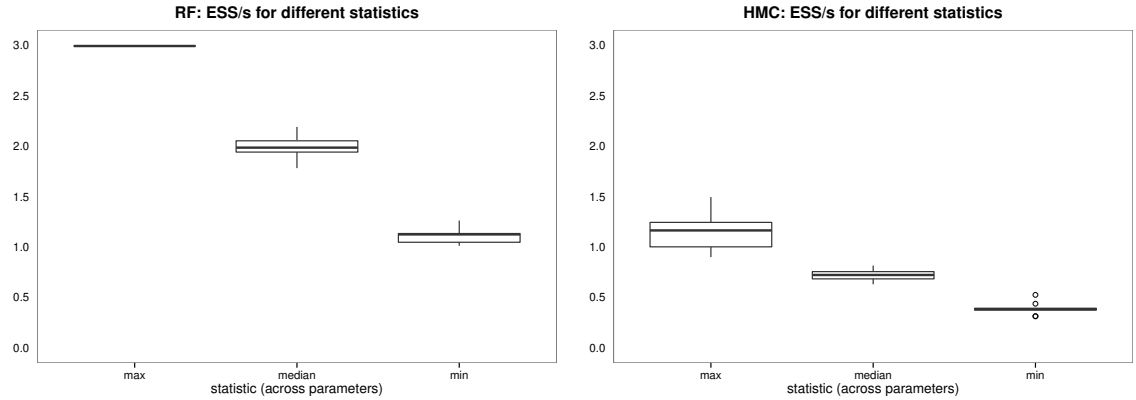

Figure 11: Boxplots of maximum, median and minimum ESS per second for BPS (left) and HMC (right).

intervals estimates from the HMC method are in close agreement with those obtained from BPS (result not shown) and both methods pass the Geweke diagnostic [8].

To compare the effectiveness of the two samplers, we look at the ESS per second of the model parameters. We show the maximum, median, and maximum over the 10 parameter components for 10 runs, for both BPS and HMC in Figure 11. We observe a speed-up by a factor two for all statistics considered (maximum, median, minimum). In the supplement, we show that the HMC chain displays much larger autocorrelations than the BPS chain.

5 Discussion

Most MCMC methods currently available, such as the MH and HMC algorithms, are discrete-time reversible processes. There is a wealth of theoretical results showing that non-reversible Markov processes mix faster and provide lower variance estimates of ergodic averages [**?**]. However, most of the non-reversible processes studied in the literature are diffusions and cannot be simulated exactly. The BPS is an alternative continuous-time Markov process which, thanks to its piecewise deterministic paths, can be simulated exactly for many problems of interest in statistics.

As any MCMC method, the BPS can struggle in multimodal scenarios and when the target exhibits very strong correlations. However, for a range of applications including sparse factor graphs, large datasets and high-dimensional settings, we have observed empirically that BPS is on par or outperforms state-of-the art methods such as HMC and Firefly. The main practical limitation of the BPS compared to HMC is that its implementation is model-specific and requires more than knowing ∇U pointwise. An important open problem is therefore whether its implementation, and in particular the simulation of bouncing times, can be fully automated. However, the techniques described in Section 2.3 are already sufficient to handle many interesting models. There are also numerous potential methodological extensions of the method to study. In particular, it has been shown in [**?**] how one can exploit the local geometric structure of the target to improve HMC and it would be interesting to investigate how this could be achieved for BPS. More generally, the BPS is a specific continuous-time piecewise deterministic Markov process [**?**]. This class of processes deserves further exploration as it might provide a whole new class of efficient MCMC methods.

Acknowledgements

Alexandre Bouchard-Côté's research is partially supported by a Discovery Grant from the National Science and Engineering Research Council. Arnaud Doucet's research is partially supported

by the Engineering and Physical Sciences Research Council (EPSRC), grants EP/K000276/1, EP/K009850/1 and by the Air Force Office of Scientific Research/Asian Office of Aerospace Research and Development, grant AFOSRA/AOARD-144042. Sebastian Vollmer's research is partially supported by the EPSRC grants EP/K009850/1 and EP/N000188/1. We thank Markus Upmeier for helpful discussions of differential geometry as well as Nicholas Galbraith, Fan Wu, and Tingting Zhao for their comments.

References

[1] R.J. Adler and J.E. Taylor. *Topological Complexity of Smooth Random Functions*, volume 2019 of *Lecture Notes in Mathematics*. Springer, Heidelberg, 2011. École d'Été de Probabilités de Saint-Flour XXXIX.

[2] R. Bardenet, A. Doucet, and C. Holmes. Towards scaling up MCMC: An adaptive subsampling approach. In *Proceedings of the 31st International Conference on Machine Learning*, 2014.

[3] P. Bratley, B.L. Fox, and L.E. Schrage. *A Guide to Simulation*. Springer-Verlag, Berlin, 1983.

[4] Michael Creutz. Global Monte Carlo algorithms for many-fermion systems. *Physical Review D*, 38(4):1228–1237, 1988.

[5] L. Devroye. *Non-uniform Random Variate Generation*. Springer-Verlag, New York, 1986.

[6] M. Einsiedler and T. Ward. *Ergodic Theory: with a view towards Number Theory*, volume 259 of *Graduate Texts in Mathematics*. Springer-Verlag, London, 2011.

[7] J.M. Flegal, J. Hughes, and D. Vats. *mcmcse: Monte Carlo Standard Errors for MCMC*, 17-08-2015. R package version 1.1-2.

[8] J. Geweke. Evaluating the accuracy of sampling-based approaches to the calculation of posterior moments. *Bayesian Statistics*, 4:169–193, 1992.

[9] M. Hairer. Convergence of Markov processes. http://www.hairer.org/notes/Convergence.pdf, 2010. Lecture notes, University of Warwick.

[10] K. Hayasaka, Takashi G., and Satoshi H. Molecular phylogeny and evolution of primate mitochondrial DNA. *Molecular Biology and Evolution*, 5:626–644, 1988.

[11] M.D. Hoffman and A. Gelman. The no-U-turn sampler: Adaptively setting path lengths in Hamiltonian Monte Carlo. *Journal of Machine Learning Research*, 15(4):1593–1623, 2014.

[12] Alan M Horowitz. A generalized guided Monte Carlo algorithm. *Physics Letters B*, 268(2):247–252, 1991.

[13] J.P. Huelsenbeck and F. Ronquist. MRBAYES: Bayesian inference of phylogenetic trees. *Bioinformatics*, 17(8):754–755, 2001.

[14] T. S. Jaakkola and M. I. Jordan. Bayesian logistic regression: a variational approach. *Statistics and Computing*, 10:25–37, 2000.

[15] T.A. Kampmann, H.H. Boltz, and J. Kierfeld. Monte Carlo simulation of dense polymer melts using event chain algorithms. *Journal of Chemical Physics*, (143):044105, 2015.

[16] J. S. Liu. *Monte Carlo Strategies in Scientific Computing*. Springer, 2008.

[17] D. Maclaurin and R.P. Adams. Firefly Monte Carlo: Exact MCMC with subsets of data. In *Uncertainty in Artificial Intelligence*, volume 30, pages 543–552, 2014.

[18] M. Manon, S.C. Kapfer, and W. Krauth. Generalized event-chain Monte Carlo: Constructing rejection-free global-balance algorithms from infinitesimal steps. *Journal of Chemical Physics*, 140(5):054116, 2014.

[19] M. Manon, J. Mayer, and W. Krauth. Event-chain Monte Carlo for classical continuous spin models. *Europhysics Letters*, (112):20003, 2015.

[20] R.M. Neal. Markov chain Monte Carlo using Hamiltonian dynamics. In *Handbook of Markov Chain Monte Carlo*. Chapman & Hall/CRC, 2011.

[21] Y. Nishikawa, M. Manon, W. Krauth, and K. Hukushima. Event-chain Monte Carlo algorithm for the Heisenberg model. *Physical Review E*, (92):063306, 2015.

[22] E.A. J. F. Peters and G. de With. Rejection-free Monte Carlo sampling for general potentials. *Physical Review E*, 85:026703, 2012.

[23] G. O. Roberts and J.S. Rosenthal. Optimal Scaling for Various Metropolis-Hastings Algorithms. *Statistical Science*, 16(4):351–367, 2001.

[24] S. Tavaré. Some probabilistic and statistical problems in the analysis of DNA sequences. *Lectures on Mathematics in the Life Sciences*, 17:56–86, 1986.

[25] Y. W. Teh, A. H. Thiéry, and S. J. Vollmer. Consistency and fluctuations for stochastic gradient Langevin dynamics. *Journal of Machine Learning Research*, 2015.

[26] V.H. Thanh and C. Priami. Simulation of biochemical reactions with time-dependent rates by the rejection-based algorithm. *Journal of Chemical Physics*, 143(5):054104, 2015.

[27] M.J. Wainwright and M.I. Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1(1-2):1–305, 2008.

[28] Z. Wang, S. Mohamed, and N. De Freitas. Adaptive Hamiltonian and Riemann manifold Monte Carlo. In *Proceedings of the 30th International Conference on Machine Learning*, pages 1462–1470, 2013.

[29] M. Welling and Y. W. Teh. Bayesian learning via stochastic gradient Langevin dynamics. In *Proceedings of the 28th International Conference on Machine Learning*, pages 681–688, 2011.

[30] T Zhao, Z. Wang, A. Cumberworth, J. Gsponer, N. De Freitas, and A. Bouchard-Côté. Bayesian analysis of continuous time Markov chains with application to phylogenetic modelling. *Bayesian Analysis*, 2015.