The Masked Bouncy Particle Sampler: A Parallel, Chromatic, Piecewise-Deterministic Markov Chain Monte Carlo Method

James Thornton
University of Oxford

George Deligiannidis
University of Oxford

Arnaud Doucet
University of Oxford

Abstract

Piecewise deterministic Markov Processes (PDMP) provide the foundation for a promising class of non-reversible, continuous-time Markov Chain Monte Carlo (MCMC) procedures and have been shown experimentally to enjoy attractive scaling properties in high-dimensional settings. This work introduces the Masked Bouncy Particle Sampler (BPS), a flexible MCMC procedure within the PDMP framework that exploits model structure and modern parallel computing resources using chromatic spatial partitioning ideas from the discrete-time MCMC literature (Gonzalez et al. 2011). We extend the basic procedure by introducing a dynamic factorisation scheme of the target distribution to reduce boundary effects commonly associated to fixed partitioning. We theoretically justify the validity of the proposed methods and provide experimental evidence that the Masked Bouncy Particle Sampler delivers significant efficiency gains over other state-of-the-art sampling schemes for certain high-dimensional sparse models.

1 INTRODUCTION

Markov Chain Monte Carlo (MCMC) procedures are a well understood and widely applied class of algorithms to perform statistical inference for complex models (Robert et al. 2013). Modern, state-of-the-art statistical and machine learning models are often high-dimensional and leverage large data-sets; most MCMC algorithms however scale poorly in such settings due to being inherently difficult to parallelise, or reliant on accept/reject steps. Such rejection procedures typically require access to all available data in the accept/reject step or prove inefficient in high-dimensional settings due to the difficulty of finding an appropriate proposal distribution.

It has been shown that existing Piecewise deterministic MCMC (PD-MCMC) procedures such as Local BPS (Peters et al. 2012; Bouchard-Côté et al. 2018), Zig-Zag (Bierkens et al. 2019), Stochastic BPS (Pakman et al. 2017) and Coordinate Sampler (Wu et al. 2018) are rejection-free and benefit from forms of sub-sampling. In addition, as with BPS, one may use gradient information in the deterministic state propagation of a PDMP procedure. Gradients are often cheap to compute using recent automatic differentiation libraries and have been shown to improve mixing dynamics in high-dimension, for example with Hamiltonian or Langevin Monte Carlo. Despite these scalable features, piecewise deterministic procedures remain inherently sequential and therefore have limited compatibility with parallel computing technologies. There have however been some successes in parallel discrete time MCMC procedures (Terenin et al. 2015; Daskalakis et al. 2018; Gonzalez et al. 2011; Song et al. 2017; Jacob et al. 2017), though most rely on rejection steps and hence it is not obvious how one may adapt these methods to rejection-free PDMP schemes with deterministic updates.

1.1 CONTRIBUTIONS AND ORGANISATION

This article introduces the Masked Bouncy Particle Sampler, a parallel PD-MCMC sampling scheme to sample from sparse graphical models. By representing the target distribution as a factor graph (Wainwright et al. 2008), the Masked BPS extends the benefits of model decomposition exhibited by the Local BPS by introducing conditional independence between groups of factors. This may be viewed as splitting the factor graph into sub-graphs. Conditional independence permits parallel updates of the parameters within each sub-graph, and hence allows the use of additional computing resources. This form of parallel computation is referred to as *chromatic*, after Gonzalez et al. 2011. Additionally, we introduce a dynamic spatial partitioning scheme within the Local BPS and Masked BPS frameworks. Dynamic partitioning of spatial regions induces a dynamic factor graph representation on the distribution of interest and hence reduces boundary effects associated with static model decomposition, as discussed in Song et al. 2017.

The structure of this paper is as follows. We first introduce general PDMPs in Section 2 and review some PDMP-based MCMC schemes that have recently been proposed. The Masked BPS methodology is then detailed in Section 3, starting with the basic algorithm in Section 3.1, followed by the dynamic factorisation scheme in Section 3.2. The proposed methodology is theoretically justified in Section 3.5. Finally, in Section 4, the efficiency of state-of-the-art samplers are compared experimentally to illustrate the benefits of the Masked BPS on sparse graphical models.

1.2 SETTING AND NOTATION

This article provides methodology for the task of sampling from a target probability distribution π with the ultimate objective of computing expectations of some class of functions with respect to the density of π, which may be evaluated up to a normalising constant, as shown in Equation (1) below. It shall be assumed throughout that π has support on $\mathcal{X} = \mathbb{R}^d$ for some dimension $d \in \mathbb{Z}^+$, and is accompanied by the usual Borel σ-algebra $\mathcal{B}(\mathbb{R}^d)$. It is also assumed that π admits a continuously differentiable density with respect to the Lebesgue measure. In an abuse of notation, this density will also be referred to as π.

We focus here on the class of distributions whereby conditional dependencies can be represented graphically and hence where the target density may be factorised as in Equation (2), though this factorisation need not necessarily be unique. Let $G = (F, N, E)$ be a bipartite, undirected, graph representing a graphical model consisting of vertices $F = \{f_1, f_2, \ldots\}$ corresponding to factors $\{\gamma_f\}_f$; vertices $N = \{1, \ldots, d\}$ corresponding to variables $\{x_i\}_i$; and, edges E connecting elements of N and F. G is referred to as a factor graph (Wainwright et al. 2008). The target π can be represented as a factor graph through the factorisation:

$$\pi(x) \propto \gamma(x), \tag{1}$$

$$\gamma(x) = \prod_{f \in F} \gamma_f(x_f), \tag{2}$$

where $x_f = \{x_i | i \in N_f\}$ is the subset of all variables x corresponding to factor f. Specifically, $N_f \subset N$, where where $i \in N_f$ if $(i, f) \in E$. Throughout this article, γ shall be expressed through its potential energy, $U(x) = -\log \gamma(x)$ and in factorised form:

$$U(x) = \sum_{f \in F} U_f(x_f), \tag{3}$$

$$U_f(x_f) = -\log \gamma_f(x_f). \tag{4}$$

For notational purposes, let $\mathcal{P}(F)$ denote the power-set of the group of factors, F, let the set of neighbouring factors to factor $f \in F$ be denoted $\bar{F}_f = \{f' | N_f \cap N_{f'} \neq \emptyset\}$, and let the joining nodes between f and f' be denoted $N_{f,f'} = N_f \cap N_{f'}$. Let $\bar{N}_f = \cup_{f' \in \bar{F}_f} N_{f',f}$ denote the set of nodes containing N_f and all neighbouring nodes, connected by a factor.

2 PIECEWISE DETERMINISTIC MARKOV CHAIN MONTE CARLO

2.1 GENERAL FRAMEWORK

PDMPs are a class of continuous time Markov process introduced by Davis; see Davis 1993 for a rigorous measure-theoretic treatment. In the interest of brevity, we will here follow the informal presentation of Vanetti et al. 2017 and Fearnhead et al. 2018 where a PDMP $\{z_t \in \mathcal{Z} | t \in \mathbb{R}^+\}$ shall be viewed as a càdlàg process that evolves as follows: events occur at times given by an inhomogeneous Poisson Process with state-dependent event rate; between events the process evolves deterministically, and at events the state of the process changes according to some transition kernel.

1. **Event Rate**
   Define event rate $\lambda : \mathcal{Z} \to \mathbb{R}^+$ where event times τ are simulated according to the inhomogeneous Poisson process: $\mathbb{P}(\tau > t) = \exp(-\int_0^t \lambda(z_s)\mathrm{d}s)$.

2. **Deterministic State Propagation**
   Let state z_t evolve according to an ordinary differential equation (ODE) with drift $\phi : \mathcal{Z} \to \mathcal{Z}$ where: $\frac{\mathrm{d}z_t}{\mathrm{d}t} = \phi(z_t)$.

3. **Event Operation**
   At event times the state transitions, $z_t \to z_t'$, according to some kernel Q, i.e. $z_t' \sim Q(\cdot | z_t)$.

We select the event rate, ODE and kernel to construct a PDMP which admits an invariant distribution φ with marginal π. For example, the samplers detailed in Section 2.2 are designed with state $z_t = (x_t, v_t) \in \mathcal{X} \times \mathcal{V}$ where $\mathcal{X} = \mathbb{R}^d$ and invariant distribution $\varphi(\mathrm{d}x, \mathrm{d}v) = \pi(\mathrm{d}x)\psi(\mathrm{d}v)$. In these examples, v_t is a vector of auxiliary variables referred to as the velocity and parametrises the deterministic linear flow $\phi(z) = (v, \mathbf{0}_d)$.

2.2 EXISTING SAMPLERS

Although there are many PDMP sampling schemes in the literature, the Bouncy Particle Sampler (BPS) (Peters et al. 2012; Bouchard-Côté et al. 2018) is one of the most well-studied theoretically (Deligiannidis et al. 2019; Durmus et al. 2018) with various extensions such as the Local BPS (Bouchard-Côté et al. 2018), Stochastic BPS (Pakman et al. 2017), and Generalised BPS (Wu et al. 2017). The Masked BPS proposed here is another variant of the BPS, taking inspiration from the Local BPS and Coordinate Sampler (Wu et al. 2018).

By representing the target distribution as a factor graph, the Local BPS uses event transition kernels corresponding to each factor, leading to localised updates on subsets of the velocity components. This is computationally cheaper

per update and allows for an efficient implementation using sub-sampling. The Coordinate Sampler also uses local updates on the state component, x, by nullifying all but one coordinate of the velocity vector, as described below.

2.2.1 Local Bouncy Particle Sampler

In the Local BPS procedure, there are two types of events: *refresh* events and *bounce* events, where each bounce event corresponds to one of the factors.

Local Bouncy Particle Sampler

1. **Event Rate**
   $\lambda(x,v) = \lambda_{ref} + \sum_{f\in F}\lambda_f(x,v)$
   $\lambda_f(x,v) = \max\{0, \langle\nabla_x U_f(x), v\rangle\}$
2. **Deterministic State Propagation**
   $\frac{\mathrm{d}x_t}{\mathrm{d}b} = v_t \qquad \frac{\mathrm{d}v_t}{\mathrm{d}b} = 0$
3. **Event Operation**

$$Q(x',v'|x,v) = \delta_x(x')\left[\frac{\lambda_{ref}}{\lambda(x,v)}\psi(v') + \sum_{f\in F}\frac{\lambda_f(x,v)}{\lambda(x,v)}\delta_{R_f(x)v}(v')\right]$$

The bounce operator $R_f(x)$ is defined as

$$\begin{aligned} R_f(x)v &= \Big(\mathbb{I} - 2\frac{\nabla U_f(x)\big(\nabla U_f(x)\big)^T}{||\nabla U_f(x)||^2}\Big)v \\ &= v - 2\frac{\langle\nabla U_f(x), v\rangle}{||\nabla U_f(x)||^2}\nabla U_f(x) \end{aligned}$$

Refresh events are triggered at the arrival times of a Poisson process of constant rate λ_{ref}. The bounce event per factor f occurs according to an inhomogeneous Poisson process of rate $\lambda_f(x,v)$, involving only x_f and v_f. By the principle of superposition of Poisson processes, one may sample event times τ with rate $\lambda(x,v)$ by sampling a bounce time proposal τ_f for each factor f using $\lambda_f(x,v)$ and a refresh time proposal τ_{ref}, and setting $\tau = \min\{\tau_{ref}, \{\tau_f\}_f\}$.

A refresh event involves sampling a new velocity vector, $v \sim \psi$ with support $\mathcal{V} = \mathbb{R}^d$. It is convenient to let ψ be the standard multivariate Gaussian distribution, as shall be done in this work, but other distributions have been considered in Bouchard-Côté et al. 2018.

A bounce event corresponding to factor f modifies only sub-component v_f of the velocity vector, according to operator $R_f(x)$. By the memoryless property, previously sampled bounce event proposal times remain valid for those other factors whose velocity components did not change, $f' \in F \setminus \bar{F}_f$. This allows for the recycling of bounce event proposal times after each event and may be efficiently implemented using a priority queue; see Bouchard-Côté et al. 2018 for full details.

Note also that the original global BPS algorithm may be recovered by treating the complete factor graph as a single factor.

2.2.2 Coordinate Sampler

In the Coordinate Sampler, only a single coordinate evolves deterministically at any time, corresponding to a velocity $v \in \mathcal{V}$, where $\mathcal{V} = \{\pm e_i | i \in \{1,\dots,d\}\}$ and $\{e_i\}_{i=1}^d$ form the standard basis for $\mathbf{R}^d$. At event times, the active velocity coordinate may change and direction may reverse. The velocity is updated to $\pm e_i$ with probability proportional to $\lambda(x, \mp e_i)$.

Coordinate Sampler

1. **Event Rate**
   $\lambda(x,v) = \lambda_{ref} + \max\{0, \langle\nabla U(x), v\rangle\}$
2. **Deterministic State Propagation**
   $\frac{\mathrm{d}x_t}{\mathrm{d}b} = v_t \qquad \frac{\mathrm{d}v_t}{\mathrm{d}b} = 0$
3. **Event Operation**
   $Q(x',v'|x,v) = \sum_{v^*\in\mathcal{V}}\frac{\lambda(x,-v^*)}{\lambda(x)}\delta_x(x')\delta_{v^*}(v')$
   $\lambda(x) = \sum_{v^*\in\mathcal{V}}\lambda(x,-v^*) = 2d\lambda_{ref} + \sum_{i=1}^d|\frac{\partial U}{\partial x_i}|$

As only a single entry of the state vector x_t evolves during the deterministic step, the Coordinate Sampler displays a Gibbs-like behaviour.

3 MASKED BOUNCY PARTICLE SAMPLER

By generalising the Coordinate Sampler's deterministic dynamics to non-zero velocity blocks, rather than coordinates, and combining it with the factor-graph decomposition, one arrives at the Masked Bouncy Particle Sampler described below.

3.1 ALGORITHM DESCRIPTION

Before describing the Masked BPS, we first introduce Boolean mask variables, $b \in \mathcal{B} \subset \{0,1\}^d$. We refer to b and $\mathcal{B}$ as a *mask* and a *mask-set* respectively. Now consider a factor graph as described in Section 1 with factors F and variable nodes $N \in \{1,\dots,d\}$. A carefully chosen mask b may be used to create a *separation*, $\kappa(b)$, on the factors F, which spatially partitions the factor graph into sub-graphs according to Definition 1 below.

Definition 1. *The separation of the group of factors F induced by mask $b \in \{0,1\}^d$ is denoted $\kappa(b) \subseteq \mathcal{P}(F)$ and satisfies for any $F, F' \in \kappa(b)$ with $F \neq F'$: $\forall f \in F \in \kappa(b), \forall f' \in F' \in \kappa(b)$ then $b_i = 0$ for all $i \in N_{f_1} \cap N_{f_2}$.*

The Masked BPS is a PMDP consisting of bounce and

refresh events, similar to the Local BPS. However, the Masked BPS also includes additional auxiliary mask variable $b \sim \rho_{\mathcal{B}}$, where $\rho_{\mathcal{B}}$ is some discrete distribution on $\mathcal{B}$. The primary difference to the Local BPS is that the deterministic flow, $\phi(z)$, may be decomposed into the latent velocity, v, and mask b, according to $\phi(x_t, v_t, b_t) = (v \odot b, \mathbf{0}_d, \mathbf{0}_d)$, where $\odot$ refers to the element-wise product. For each $i \in \{1, \ldots, d\}$, latent velocity element v_i is said to be *masked* if $b_i = 0$. The effect of masking multiple velocity components is similar in spirit to the Coordinate Sampler's approach, but generalised to blocks rather than coordinates.

Refresh events occur at constant rate λ_{sync}, and correspond to events with transition kernel $Q_{sync}(x', v', b'|x, v, b) = \delta_x(x')\psi(v')\rho_{\mathcal{B}}(b')$. Bounce events occur per factor $f \in F$, at rate $\lambda_f(x, v \odot b)$ and correspond to transition events $Q_f(x', v', b'|x, v, b) = \delta_x(x')\delta_b(b')\delta_{R_f(x,b)v}(v')$. As described above, the active state evolves with linear rate between refresh and bounce events. Here active state refers to those coordinates of x which are not masked.

Under the described process, the state of the PDMP is $z_t = (x_t, v_t, b_t) \in \mathbb{R}^d \times \mathbb{R}^d \times \mathcal{B}$.

1. **Event Rate**
   $\lambda(x, b \odot v) = \lambda_{sync} + \sum_{f \in F} \lambda_f(x, b \odot v)$
   $\lambda_f(x, b \odot v) = \max\{0, \langle \nabla_x U_f(x), b \odot v \rangle\}$
2. **Deterministic State Propagation**
   $\frac{\mathrm{d}x_t}{\mathrm{d}b} = b_t \odot v_t \qquad \frac{\mathrm{d}v_t}{\mathrm{d}b} = 0 \qquad \frac{\mathrm{d}b_t}{\mathrm{d}b} = 0$
3. **Event Operation**
   $Q(x', v', b'|x, v, b) = \frac{\lambda_{sync}}{\lambda(x, b \odot v)} \delta_x(x')\psi(v')\rho_{\mathcal{B}}(b') + \sum_{f \in F} \frac{\lambda_f(x, b \odot v)}{\lambda(x, b \odot v)} \delta_x(x')\delta_b(b')\delta_{R_f(x,b)v}(v')$

The bounce operator $R_f(x, b)$ is constructed such that masked latent velocity components remain unchanged at bounce events. This permits bounce events corresponding to factors in different sub-graphs to occur in parallel, between refresh events. At bounce events the latent velocity vector transitions according to bounce operator $R_f(x, b)$ where $v \to R_f(x, b)v$:

$$R_f(x, b) = \Big(\mathbb{I} - 2\frac{(\nabla U_f(x) \odot b)(\nabla U_f(x) \odot b)^T}{||(\nabla U_f(x) \odot b)||^2}\Big), \tag{5}$$

$$R_f(x, b)v = v - 2\frac{\langle \nabla U_f(x) \odot b, v \rangle}{||\nabla U_f(x) \odot b||^2} \nabla U_f(x) \odot b. \tag{6}$$

Consider the factor graph under mask b with corresponding separation of factors: $\kappa(b) = \{F_1, F_2, \ldots\}$. At a bounce event corresponding to factor $f_1 \in F_1$, the operator $R_{f_1}(x, b)$ will update only non-masked v_i $i \in \{j | b_j = 1\} \cap N_{f_1}$. As a consequence of Definition 1 and of subsampling, only bounce-time proposals for factors $f' \in F_1$ whose velocity components have changed would need to be re-sampled. This means that the global factor graph may be split into sub-graphs, one for each $F_i \in \kappa(b)$. Therefore, the process may be run in parallel between refresh events of b, which acts as a synchronisation between separate worker processes for each sub-graph. This is analogous to the Chromatic sampler of Gonzalez et al. 2011, whereby non-masked variables are considered one colour and masked variables another. The Masked BPS may be implemented using Algorithm 1.

Algorithm 1: Masked Bouncy Particle Sampler

Initialization:
  $T_{sync} \sim Exp(\lambda_{sync})$, $T_{\text{prev sync}} \leftarrow 0$;
  $b \sim \rho_{\mathcal{B}}$, $v \sim \psi$, Initialize x, t ;
while *another update requested* **do**
  Parallel Bounce Events:
  foreach F_b *in* $\kappa(b)$ *in parallel* **do**
    $Q_{F_b} \sim \text{PriorityQueue}(x, v \odot b, T_{\text{prev sync}}, F_b)$;
    $(T, f) \leftarrow \text{pop Q}_{F_b}$;
    while $T < T_{sync}$ **do**
      foreach $f' \in \bar{F}_{bf}$ **do**
        $x_{f'} \leftarrow x_{f'} + b_{f'} \odot v_{f'} \odot (T - t_{f'})$;
      end
      $v_f \leftarrow R_f(x_f, b_f)(v_f)$;
      foreach $f' \in \bar{F}_{bf}$ **do**
        $\tau_{f'} \sim NewBounce(x_{f'}, b_{f'} \odot v_{f'})$;
        Update bounce time in Q_{F_b} associated to f' to the value $T + \tau_{f'}$;
        $t_{f'} \leftarrow T$
      end
      $(T, f) \leftarrow \text{pop Q}_{F_b}$;
    end
    foreach f' *in* F_b *in parallel* **do**
      $x_{f'} \leftarrow x_{f'} + b_{f'} \odot v_{f'} \odot (T_{sync} - t_{f'})$;
      $t_{f'} \leftarrow T_{sync}$;
    end
  end
  Refresh / Synchronisation event:
    $b \sim \rho_{\mathcal{B}}$;
    $v \sim \psi$;
    $\tau_{sync} \sim Exp(\lambda_{sync})$;
    $T_{\text{prev sync}} \leftarrow T_{sync}$;
    $T_{sync} \leftarrow T_{sync} + \tau_{sync}$
end

Algorithm 2: NewBounce

For given $U(\cdot), x, v$;
Simulate τ such that:
$\mathbb{P}(\tau > t) = \exp(-\int_0^t \max\{0, \langle \nabla U(x + v \odot t), v \rangle\} \mathrm{d}s)$;
Return τ;

Algorithm 3: PriorityQueue

For given T, F, x, v;
Initialize empty queue Q;
foreach $f \in F$ **do**
  $\tau_f \sim NewBounce(x_f, v_f)$;
  $T_f \leftarrow T + \tau_f$;
  Add (T_f, f) to Q in increasing order of T_f;
  Return Q
end

During refresh events, the variables b, v and τ_{sync} are sampled independently of the state of the process. This means that for a fixed pseudo time limit T, one may sample $\{(\tau_{sync,1}, v_1', b_1), (\tau_{sync,2}, v_2', b_2), \ldots, (\tau_{sync,N}, v_N', b_N)\}$ such that $\sum_i^N \tau_i > T$, as part of initialisation. In a distributed computing architecture where each factor is associated to a separate worker process, these pre-computed samples may be passed to each worker in advance of the bounce operations that run in parallel. This may reduce global communication beyond the initial sampling stage to communication between only worker processes that are associated to neighbouring factors.

3.2 DYNAMIC FACTORISATION

The performance of the Masked BPS is sensitive to the choice of mask-set, $\mathcal{B}$, and choice of factor graph decomposition F. Instead of fixing the model factorisation, one may exploit the doubly stochastic PDMP formulation detailed in Pakman et al. 2017 and Vanetti et al. 2017 to allow for a dynamic factorisation of the Local and Masked BPS.

Consider a finite collection of possible factorisations, $\mathcal{F}$, of the target density π, equipped with measure $\mu_{\mathcal{F}}$. One may express the joint distribution of x and F as follows:

$$\pi(x, F) = \pi(x|F)\mu_{\mathcal{F}}(\{F\}) \tag{7}$$

$$= \mu_{\mathcal{F}}(\{F\}) \prod_{F_l \in \mathcal{F}} \Big[\prod_{f \in F_l} \pi_f(x_f) \Big]^{\mathbb{I}\{F_l = F\}} \tag{8}$$

$$= \mu_{\mathcal{F}}(\{F\}) \prod_{f \in F} \pi_f(x_f) \tag{9}$$

It is easy to show that the factorisation may be marginalised to recover the target distribution.

$$\int \pi(x|F)\mu_{\mathcal{F}}(\mathrm{d}F) = \sum_{F \in \mathcal{F}} \mu_{\mathcal{F}}(\{F\}) \prod_{f \in F} \pi_f(x_f)$$
$$= \pi(x) \sum_{F \in \mathcal{F}} \mu_{\mathcal{F}}(\{F\}) = \pi(x).$$

In order to preserve the efficiency gains from sub-sampling in the Local BPS and Masked BPS, it is convenient to perform the factorisation transitions at refresh or synchronisation events. Indeed, given that the choice of mask-set $\mathcal{B}$ is dependent on factorisation F, one may choose to sample both F and $\mathcal{B}$ jointly at synchronisation events. This leads to an algorithm very similar to Algorithm 1, with the addition of sampling F at synchronisation times.

3.3 ILLUSTRATIVE EXAMPLES

3.3.1 Isotropic Gaussian

Figure 1 illustrates how the Masked BPS exhibits similar dynamics to both the Local BPS and Coordiante sampler. Here the samplers are shown for a simple bivariate isotropic Gaussian target distribution, where for the Masked BPS either one or none of the coordinates may be masked.

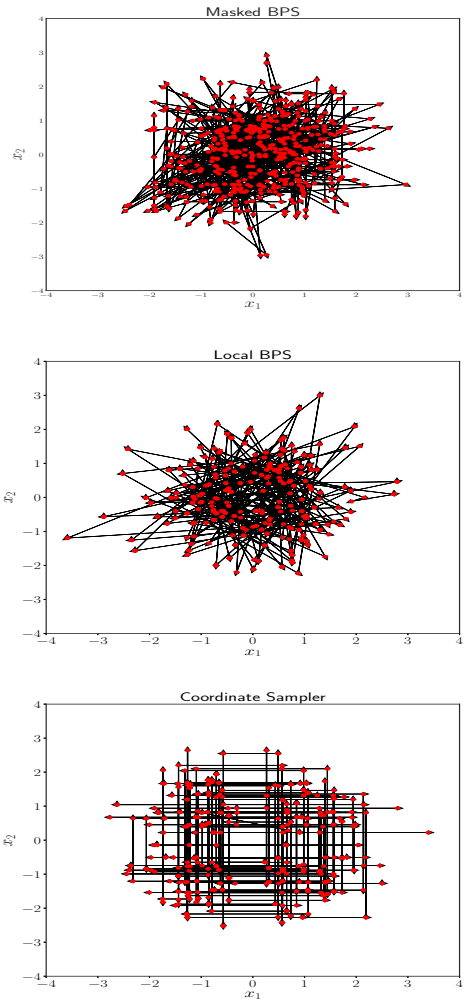

Figure 1: Path trajectory of Masked BPS, Local BPS and Coordinate Sampler for 500 events targeting an Isotropic Gaussian. Location at events in red.

3.3.2 Hierarchical Model

Figure 2 depicts a simple $d = 4$ dimensional hierarchical model with a global variable x_0 and $d - 1$ local variables (x_1, x_2, x_3). Given some dominating mea-

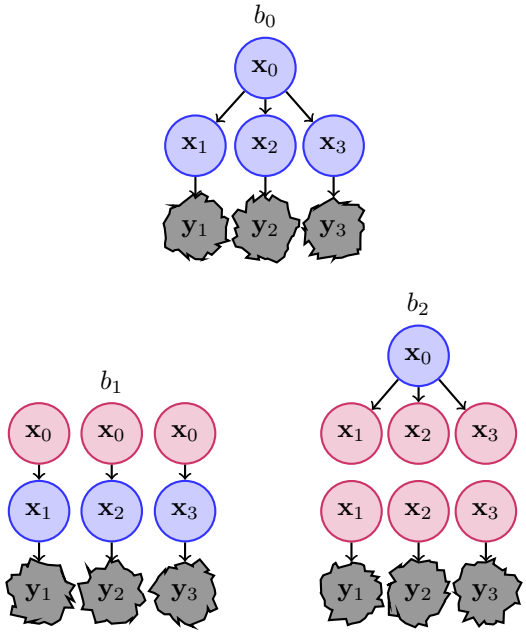

Figure 2: Blue indicates non-masked parameters, red indicates masked parameters, and grey indicates constants i.e. data when sampling a posterior. Masks: $b_0 = (1, 1, 1, 1)$, $b_1 = (0, 1, 1, 1)$, $b_2 = (1, 0, 0, 0)$

sure, this model has density π which may be factorised as: $\pi(x_0, x_1, x_2, x_3) = \pi(x_0) \prod_{i=1}^{3} \pi(x_i|x_0)\pi(y_i|x_i)$ and hence may be expressed as a factor graph with the following factors: f_0 corresponding to $\pi(x_0)$; factors f_1, f_2, f_3 corresponding to $\{\pi(x_i|x_0)\}_{i=1}^{3}$ and factors f_4, f_5 and f_6 corresponding to the observation likelihoods $\{\pi(y_i|x_i)\}_{i=1}^{3}$. The mask $b_1 = (0, 1, 1, 1)$ creates a separation between groups of factors $\kappa(b_1) = \{\{f_0\}, \{f_1, f_4\}, \{f_2, f_5\}, \{f_3, f_6\}\}$. In such a process, variable x_0 would remain stationary whilst pairs (x_1, x_2, x_3) may each explore the state space independently and in parallel. Similarly, mask $b_2 = (1, 0, 0, 0)$ induces the same partition of the factor graph but with x_0 free to explore the state space and fixed x_1, x_2, x_3.

3.3.3 Chain Model

There may be situations where sampling performance is sensitive to factorisation. For example, consider a simple chain model of length $d = 7$, depicted in Figure 3.

One may split the chain into three factors using factorisation $F_1 = (f_1, f_2, f_3)$, corresponding to groups of parameters (x_0, x_1, x_2), (x_2, x_3, x_4) and (x_4, x_5, x_6) respectively. Similarly, one may decompose the factor graph into two factors $F_2 = (f_1', f_2')$ corresponding to $x_{0:3}$ and $x_{3:6}$. To induce conditionally independent updates, one may mask parameters x_2 and x_4 under factorisation F_1, and parameter x_3 under F_2. Permutations of masking x_2 and x_4 recovers separations labelled 1 an 2 in Figure 3 and masking x_3

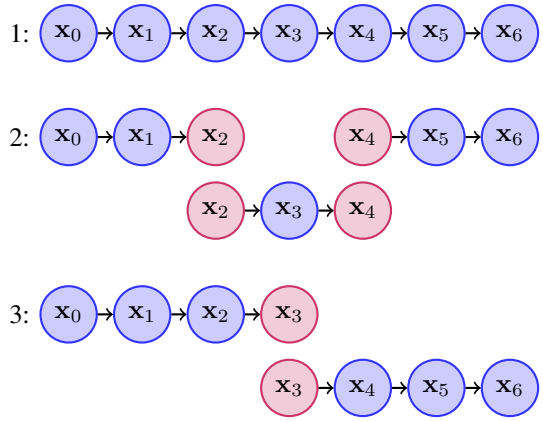

Figure 3: Diagram of dimension $d = 7$ Gaussian chain model, with example separations into sub-graphs under different factorisation schemes.

may result in splits labelled 1 and 3 for example. If there is strong inter-dependency between x_2 and x_1 (or x_4 and x_5), then mixing will be slow under factorisation F_1 as x_2 (and/ or x_4) will be static for periods. However, mixing may be faster under factorisation F_2. If the dependency is not known a priori, then one may use a dynamic factorisation to reduce the risk of choosing an inefficient factorisation. Indeed one may view choosing the factorisation, or collection of factorisations, in the same way as tuning a hyper-parameter to improve algorithm performance.

3.4 DISCUSSION

3.4.1 Related Samplers

The mask-set $\mathcal{B}$, factorisation-set $\mathcal{F}$, and sampling distributions ψ, $\rho_{\mathcal{B}}$ and $\mu_{\mathcal{F}}$ may be considered tuning parameters in the Masked BPS procedure. Consequently, one may construct a wide class of samplers with such choices.

For example, for any given factorisation F, one may choose a mask-set consisting only of 1s i.e. $\mathcal{B}_1 = \{(1, 1, \ldots, 1)\}$ hence recovering the Local BPS. Similarly, one may choose the trivial factorisation-set consisting of only a single factor for the whole model and the trivial mask-set consisting of 1s to recover the regular BPS. Indeed, by choosing the factorisation set to include both the trivial single factor factorisation in addition to a non-trivial factorisation, one may switch between the BPS and Local BPS at each refresh event according to distribution $\mu_{\mathcal{F}}$.

The Masked BPS interpolates between the Local BPS and Coordinate Sampler. By choosing the mask-set to be those masks with 0 entries except a single coordinate and choosing the factorisation set of a single factor, one recovers a Coordinate Sampler style procedure. The primary difference to the Coordinate Sampler being that there are bounce events which transitions velocity $v \odot e_i \to -v \odot e_i$ under operator $R_f(x, e_i)$ between refresh events. Hence with the

Masked BPS under such specification, the active coordinate may only change at refresh events, rather than switching at each event as with the Coordinate Sampler. This is likely to lead the Masked BPS to be less efficient than the Coordinate Sampler, when using such choices of mask and factorisation set.

3.4.2 Limitations

The Masked BPS allows one to exploit parallel computing resources, this however comes at the cost of fixing the state of masked coordinates in the PDMP, as seen in the trace plot Figure 6. Similar to the Coordinate Sampler, holding components of the state constant through time will lead to increased auto-correlation of those components and slower mixing of the process relative to non-masked parameters. Additionally, if the Masked BPS is implemented to run in parallel, there is a communication cost to also consider. One must therefore balance the benefit of being able to use additional computing resources with the downsides of a less efficient PDMP per iteration and increased communication cost. This balance may be managed through careful tuning of the mask-set. However, this makes the parallel capabilities of the Masked BPS primarily suited to the case of very large models whereby operations involving the whole model are sufficiently expensive to warrant splitting the model.

Although a dynamic factorisation may hedge against a poor choice of factorisation, an unfortunate consequence of having multiple factorisations is that one must be able to sample bounce event proposal times for each factor in each factorisation. There are tools to assist sampling event times such as thinning, superposition and time-scale transformation methods, see Bouchard-Côté et al. 2018. However, efficiently sampling exact event times is in general not trivial given that each event time proposal is specific to the model and factor. Pakman et al. 2017 suggests using a general predictive model in combination with thinning in order to automatically simulate event times. Although this may reduce the requirement of specifying bounce event proposals by the user, the proposed method is biased.

3.5 THEORETICAL RESULTS

In this section, we establish that the Masked BPS admits an invariant distribution with marginal π.

For ease of notation, let refresh events correspond to index 0 and bounce events be indexed by $i \in [m] = \{1, \dots, m\}$ correspond to factors $f \in F$, for some bijection between $[m]$ and F. Hence the dynamics of the Masked BPS described in Section 3 may be specified in terms of flow ϕ with kernels and event-rates $\{Q_i, \lambda_i\}_{i=0}^m$.

Lemma 3.1. *The Masked BPS dynamics detailed in Section 3 satisfy the following conditions of Vanetti et al. 2017.*

1. *Mapping $\mathcal{S} : \mathcal{Z} \to \mathcal{Z}$ where $\mathcal{S}(x, v, b) = \mathcal{S}(x, -v, b)$ is φ-preserving, hence:*

$$\varphi(\mathrm{d}z) = \varphi(\mathcal{S}(\mathrm{d}z)).$$

2. *Event rates $\{\lambda_i\}_i$ satisfy:*

$$\sum_{i=0}^{m} [\lambda_i(\mathcal{S}(z)) - \lambda_i(z)] = \nabla \cdot \phi(z) - \langle \nabla U(z), \phi(z) \rangle.$$

3. *For each $i \in \{0, 1, \dots, m\}$:*

$$\int \lambda_i(z) Q_i(z, \mathrm{d}z') \varphi(\mathrm{d}z) = \varphi(\mathcal{S}^{-1}(\mathrm{d}z')) \lambda_i(\mathcal{S}(z'))$$

These conditions are verified in the supplementary material.

Lemma 3.2. *The generator $\mathcal{L}$ for the Masked BPS is given for functions $h : \mathbb{R}^d \times \mathbb{R}^d \times \mathcal{B} \to \mathbb{R}$ by:*

$$\begin{aligned}\mathcal{L}h(x,v,b) = &\\ &\langle \nabla_x h(x, v, b), b \odot v \rangle \\ &+ \sum_{f \in F} \lambda_f(x, b \odot v) \{h(x, R_f(x, b)v, b) - h(x, v, b)\} \\ &+ \lambda_{sync} \int \{h(x, v', b') - h(x, v, b)\} \psi(\mathrm{d}v') \rho_{\mathcal{B}}(\mathrm{d}b').\end{aligned}$$

Here h is bounded and satisfies the regularity conditions given by (Davis 1993, Theorem 26.14).

Theorem 3.1. *The Masked BPS described above and by Algorithm 1 induces a PDMP with invariant extended distribution $\varphi(\mathrm{d}x, \mathrm{d}v, \mathrm{d}b) = \pi(\mathrm{d}x)\psi(\mathrm{d}v)\rho_{\mathcal{B}}(\mathrm{d}b)$, and hence desired marginal distribution π.*

If $\int \mathcal{L}h(x, v, b)\pi(\mathrm{d}x)\psi(\mathrm{d}v)\rho_{\mathcal{B}}(\mathrm{d}b) = 0$ then by (Davis 1993, Theorem 24.7) the procedure admits the correct invariant distribution. The proof is detailed fully in the supplementary material. The result also follows directly from Proposition 2 of Vanetti et al. 2017, given Lemma 3.1 detailed earlier.

Lemma 3.3. *The generator $\mathcal{L}$ for the Masked BPS with dynamic factorisation is given by:*

$$\begin{aligned}\mathcal{L}h(x,v,b,F) = &\\ &\langle \nabla_x h(x, v, b, F), b \odot v \rangle \\ &+ \sum_{f \in F} \lambda_f(x, b \odot v) \{h(x, R_f(x, b)v, b, F) - h(x, v, b, F)\} \\ &+ \lambda_{sync} \int \{h(x, v', b', F') - h(x, v, b, F)\} \\ &\qquad \psi(\mathrm{d}v') \rho_{\mathcal{B}_F}(\mathrm{d}b') \mu_{\mathcal{F}}(\mathrm{d}F')\end{aligned}$$

Theorem 3.2. *The Masked BPS with Dynamic Factorisation induces a PDMP with invariant extended distribution $\varphi(\mathrm{d}x, \mathrm{d}v, \mathrm{d}b, dF) = \pi(\mathrm{d}x)\psi(\mathrm{d}v)\rho_{\mathcal{B}_F}(\mathrm{d}b)\mu_{\mathcal{F}}(\mathrm{d}F)$, and hence desired marginal distribution π.*

A direct proof using the generator is given in the supplementary material. This may also be shown using Lemma 3.1 to verify conditions of Proposition 3 in Vanetti et al. 2017 for doubly stochastic PDMPs.

A refresh event for auxiliary variables b and v at synchronisation times is sufficient to make the process irreducible, providing that after each refresh there is a positive probability of each velocity component being non-masked. Hence, by Davis 1993, an irreducible PDMP which targets the invariant distribution π may be used to compute integrals with respect to π.

4 NUMERICAL EXPERIMENTS

4.1 IMPLEMENTATION DETAILS

The Masked BPS [1] will be compared against the Local BPS and manually tuned Hamiltonian Monte Carlo (HMC), implemented in the *hamiltorch* package by Cobb et al. 2019. In order to leverage the parallel features of the Masked BPS, sub-graphs are initiated within worker processes at each refresh step using *ray* (Moritz et al. 2018). The deterministic step is then ran in parallel. Communication between workers and a central node is performed using Apache arrow[2] (Lentner 2019). In order to avoid overhead from communication, each worker serialises output from the MCMC iterations between synchronisation and only communicates the most recent state of the sub-graph at synchronisation. Sampling algorithms were executed for a fixed time budget and compared using effective sample size (ESS) implemented in the *arviz* package (Kumar et al. 2019).

4.2 CHAIN GAUSSIAN MARKOV RANDOM FIELD

Consider the toy example of a chain-shaped distribution of length d with conditional independence between non-adjacent nodes, as shown in equation (10) and depicted graphically in Figure 3 for length $d = 7$.

$$\pi(x) = \pi(x_0) \prod_{k=1}^{d-1} \pi(x_k | x_{k-1}) \tag{10}$$

It is clear this is a Gaussian Markov Random Field (MRF), with tri-diagonal precision matrix Λ with pairwise precision ρ, between adjacent nodes. Hence $x \sim \mathcal{N}(0, \Lambda^{-1})$.

To demonstrate how the Masked BPS may be used, consider the case where $d = 7$ the density is be expressed as three factors with one node shared between adjacent factors and pairwise precision set to $\rho = 0.5$. Hence $\pi(x) = \pi(x_{0:2})\pi(x_{3:4}|x_2)\pi(x_{5:6}|x_4)$. The sampler was set with refresh rate 0.01 and a single node x_2 or x_4 was masked at each refresh, with equal probability. Full details are given in the supplementary material.

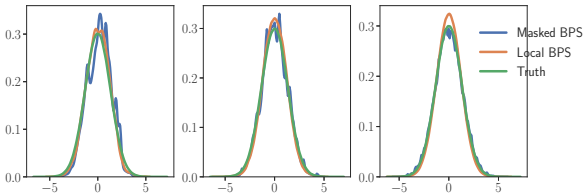

Figure 4: Evolution of x_2 density after $3s$, $30s$ and $300s$ of the same simulation run.

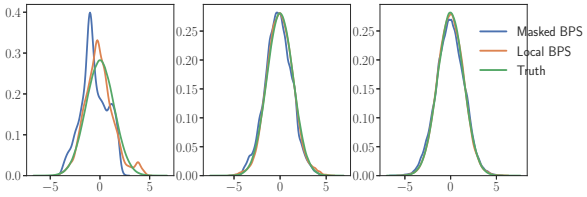

Figure 5: Evolution of x_1 density after $0.3s$, $3s$ and $30s$ of the same simulation run.

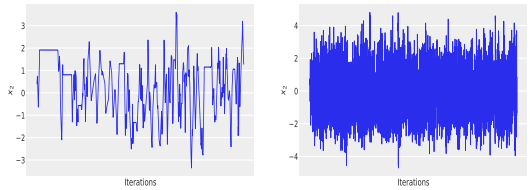

Figure 6: Trace of x_2. Left plot is a zoomed-in view of initial segment of the right plot.

It can be seen from Figures 4 and 5 that, as expected, the nodes with positive probability of being masked converge slower than those not masked. Node x_2 is periodically fixed. However, for a carefully chosen refresh rate, this becomes inconsequential after sufficient iterations. This is illustrated by Figure 6, where it is difficult to visually identify the fixed segments in the full trace on the right.

Now consider a more challenging toy example of sampling from a chain-shaped Markov Random Field of significantly higher dimension. This is less trivial because operations involving a large number of parameters are computationally intensive.

In the following experiment, chain models were constructed in blocks: the first block consisting of 100 nodes and subsequent blocks of 100 nodes, with an overlap of a single node. This experiment was ran for chain models of n factors where $n \in \{10, 25, 50, 100, 150\}$ corresponding

[1] Implementation: `https://github.com/anon284/Masked_BPS_UAI_2020`

[2] `https://arrow.apache.org`

to a total number of 991; 2476; 4951; 9901 and 14, 851 parameters respectively. A total of 30 worker processes were utilised and the mask-set was chosen to be the collection of vectors with up to 29 masked coordinates joining factors, hence leading to a maximum of 30 sub-graphs being generated. The mask was sampled uniformly on this.

It can be seen from Table 1 that the Masked BPS outperforms the other methods considered in terms of average ESS. Significantly more iterations occur under the Masked BPS procedure than Local, due to parallel updates and the extra computational resources employed. However, the ratio of ESS / iteration is lower as per-iteration updates are less efficient. This is as expected due to some of the nodes being held constant periodically. As the number of factors increases beyond the number of worker processes, the performance gap between the samplers narrows.

Table 1: ESS, averaged over execution time for the Gaussian Chain MRF

	ESS / s				
n	10	25	50	100	150
Masked BPS	**48.21**	**40.62**	**22.65**	**8.16**	**5.97**
Local BPS	31.15	11.80	6.451	2.87	1.41
HMC	0.78	0.31	0.167	0.07	0.05

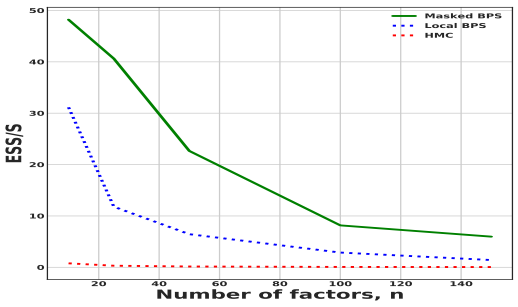

Figure 7: ESS per second for various number of factors for chain MRF

4.3 HIERARCHICAL GAUSSIAN MARKOV RANDOM FIELD

In a simple hierarchical model, similar to that depicted in Figure 2, let $d+1$ denote the number of variables $x_{0:d}$ in a model where there is one global variable, x_0, and d local variables. This may be considered a toy model for the situation whereby one wishes to model separate groups of observations using within-group parameters and also parameters shared across groups.

One may factorise the density as:

$$\pi(x_0)\prod_{i=1}^{d}\pi(x_i|x_0) \tag{11}$$

For simplicity, let $\pi(x_0)$ be a Gaussian density and $\pi(x_i|x_0)$ the conditional Gaussian density. Jointly this forms a Gaussian MRF with pair-wise precision ρ between x_0 and each x_i, $i > 0$.

Table 2: Hierarchical Gaussian Model with pairwise precision $\rho = 0.5$ with 46 worker processes for the Masked BPS

	ESS/s			
d	10	45	100	150
Masked BPS	29.5	**4.98**	**0.98**	**0.45**
Local BPS	**38.4**	1.93	0.48	0.03
HMC	1.83	0.37	0.16	0.02

The results in Table 2 tell a similar story to those in the previous experiments – at higher dimensions, the Masked BPS outperforms the other samplers considered in terms of ESS.

5 CONCLUSION AND EXTENSIONS

This work introduces the Masked BPS and a dynamic factorisation scheme to improve performance of the sampler. The Masked BPS is shown to be highly flexible, and shares similar properties to other reversible and non-reversible sampling procedures. It has been shown in (Wu et al. 2018) and in (Deligiannidis et al. 2019; Durmus et al. 2018) respectively that the Coordinate Sampler and BPS exhibit exponential ergodicity under weak assumptions on the target distribution. Given that the Masked BPS generalises the Local BPS, BPS and has similar dynamics to the Coordinate Sampler, it may be the case that the Masked BPS also admits such properties. This is left to future work.

In addition, it may not be obvious how to choose the factorisation-set and mask-set for many factor-graphs. Further work is required to better understand appropriate graph splitting techniques and how performance is related to graph topology.

Although the Masked BPS involves distributed computation of events, the procedure is not asynchronously parallel as a barrier is required in order to perform refresh events and change the Boolean mask. This involves synchronising worker processes. Recent developments by Terenin et al. 2015; Daskalakis et al. 2018; Feng et al. 2019 provide methodology to perform asynchronous Gibbs sampling in discrete time. A natural extension would be to investigate whether such asynchronous methods could be applied in the piecewise deterministic setting.

References

Bierkens, Joris, Paul Fearnhead, Gareth Roberts, et al. (2019). "The zig-zag process and super-efficient sampling for Bayesian analysis of big data". In: *The Annals of Statistics* 47.3, pp. 1288–1320.

Bouchard-Côté, Alexandre, Sebastian J Vollmer, and Arnaud Doucet (2018). "The bouncy particle sampler: A nonreversible rejection-free Markov chain Monte Carlo method". In: *Journal of the American Statistical Association* 113.522, pp. 855–867.

Cobb, Adam D et al. (2019). "Introducing an Explicit Symplectic Integration Scheme for Riemannian Manifold Hamiltonian Monte Carlo". In: *arXiv preprint arXiv:1910.06243*.

Daskalakis, Constantinos, Nishanth Dikkala, and Siddhartha Jayanti (2018). *HOGWILD!-Gibbs can be PanAccurate*. arXiv: 1811.10581 [cs.LG].

Davis, Mark HA (1993). *Markov models & optimization*. Routledge.

Deligiannidis, George, Alexandre Bouchard-Côté, Arnaud Doucet, et al. (2019). "Exponential ergodicity of the bouncy particle sampler". In: *The Annals of Statistics* 47.3, pp. 1268–1287.

Durmus, Alain, Arnaud Guillin, and Pierre Monmarché (2018). "Piecewise Deterministic Markov Processes and their invariant measure". In: *arXiv preprint arXiv:1807.05421*.

Fearnhead, Paul et al. (2018). "Piecewise deterministic Markov processes for continuous-time Monte Carlo". In: *Statistical Science* 33.3, pp. 386–412.

Feng, Weiming, Thomas P Hayes, and Yitong Yin (2019). "Fully-Asynchronous Distributed Metropolis Sampler with Optimal Speedup". In: *arXiv preprint arXiv:1904.00943*.

Gonzalez, Joseph et al. (2011). "Parallel gibbs sampling: From colored fields to thin junction trees". In: *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pp. 324–332.

Jacob, Pierre E, John O'Leary, and Yves F Atchadé (2017). "Unbiased markov chain monte carlo with couplings". In: *arXiv preprint arXiv:1708.03625*.

Kumar, Ravin et al. (2019). "ArviZ a unified library for exploratory analysis of Bayesian models in Python". In: *The Journal of Open Source Software*. DOI: 10.21105/joss.01143. URL: http://joss.theoj.org/papers/10.21105/joss.01143.

Lentner, Geoffrey (2019). "Shared Memory High Throughput Computing with Apache Arrow™". In: *Proceedings of the Practice and Experience in Advanced Research Computing on Rise of the Machines (learning)*, pp. 1–2.

Moritz, Philipp et al. (2018). "Ray: A Distributed Framework for Emerging AI Applications". In: *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*. OSDI'18. Carlsbad, CA, USA: USENIX Association, pp. 561–577. ISBN: 9781931971478.

Pakman, Ari et al. (2017). "Stochastic bouncy particle sampler". In: *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, pp. 2741–2750.

Peters, Elias AJF et al. (2012). "Rejection-free Monte Carlo sampling for general potentials". In: *Physical Review E* 85.2, p. 026703.

Robert, Christian and George Casella (2013). *Monte Carlo statistical methods*. Springer Science & Business Media.

Song, Jun and David Moore (2017). "Parallel Chromatic MCMC with Spatial Partitioning". In: *Workshops at the Thirty-First AAAI Conference on Artificial Intelligence*.

Terenin, Alexander, Daniel Simpson, and David Draper (2015). "Asynchronous gibbs sampling". In: *arXiv preprint arXiv:1509.08999*.

Vanetti, Paul et al. (2017). "Piecewise-Deterministic Markov Chain Monte Carlo". In: *arXiv preprint arXiv:1707.05296*.

Wainwright, Martin J, Michael I Jordan, et al. (2008). "Graphical models, exponential families, and variational inference". In: *Foundations and Trends® in Machine Learning* 1.1–2, pp. 1–305.

Wu, Changye and Christian P Robert (2017). "Generalized bouncy particle sampler". In: *arXiv preprint arXiv:1706.04781*.

– (2018). "The Coordinate Sampler: A Non-Reversible Gibbs-like MCMC Sampler". In: *arXiv preprint arXiv:1809.03388*.

6 SUPPLEMENTARY MATERIAL

6.1 SUBSAMPLING

Recall that the proposal times τ_fs follow the rule: $\mathbb{P}(\tau_f > t) = \exp\left\{-\int_0^t \max\{0, \langle \nabla U_f(x + v \odot t), v\rangle\}\mathrm{d}s\right\}$. For a fixed velocity vector, v_f, the sampled event times from a Poisson process exhibits a form of memorylessness:

$$\begin{aligned}\mathbb{P}(\tau_f > t + T | \tau_f > T) &= \frac{\exp(-\int_0^{t+T} \max\{0, \langle \nabla U_f(x + v \odot t), v\rangle\}\mathrm{d}s)}{\exp(-\int_0^{T} \max\{0, \langle \nabla U_f(x + v \odot t), v\rangle\}\mathrm{d}s)} \\ &= \exp(-\int_0^{t} \max\{0, \langle \nabla U_f(x + v \odot t), v\rangle\}\mathrm{d}s) \\ &= \mathbb{P}(\tau_f > t).\end{aligned}$$

6.2 ALGORITHMS

Algorithm 4: Local BPS via Priority Queue Implementation

Given x, v, F and $U_f, \nabla U_f$ for $f \in F$;
Initialization:
    Initialize x, t and $v \sim \psi$;
    $T_{ref} \sim Exp(\lambda_{ref})$;
    $T \leftarrow 0$;
    Q $\sim$ PriorityQueue(x,v,T,F)
foreach $i \in \{1, \ldots\}$ **do**
    $(T, f) \leftarrow$ pop Q
    if $T < T_{ref}$ **then**
        foreach $f' \in \bar{F}_f$ **do**
            $x_{f'} \leftarrow x_{f'} + v_{f'} \odot (T - t_{f'})$;
        end
        $v_f \leftarrow R_f(x_f)(v_f)$;
        foreach $f' \in \bar{F}_f$ **do**
            $\tau_{f'} \sim NewBounce(x_{f'}, v_{f'})$;
            Update bounce time in Q associated to f' to the value $T + \tau_{f'}$;
            $t_{f'} \leftarrow T$
        end
    end
    else
        $x' \leftarrow x + v \odot (T_{ref} - t)$;
        $v' \sim \psi$;
        Q $\sim$ PriorityQueue(x',v',T_{ref}, F);
        $\tau_{ref} \sim Exp(\lambda_{ref})$;
        $T^m_{ref} \leftarrow T^m_{ref} + \tau_{ref}$;
        $x \leftarrow x', v \leftarrow v', t \leftarrow T_{ref}$;
    end
end

6.3 PROOFS OF THEORETICAL RESULTS

Lemma 6.1. *The bounce operation, $R_f(x,b)$, on the masked process enjoys the following properties:*

$$||R_f(x,b)v|| = ||v||, \tag{12}$$
$$R_f(x,b)R_f(x,b) = \mathbb{I}, \tag{13}$$
$$\langle \nabla_x U_f(x), b \odot R_f(x,b)v\rangle = -\langle \nabla_x U_f(x), b \odot v\rangle. \tag{14}$$

Proof. Each property is proved separately:

- $||R_f(x,b)v|| = ||v||$:
  By definition, one has

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

  Hence, by simple expansion, we obtain

$$\begin{aligned}||R_f(x,b)v||^2 &= ||v||^2 - 4\frac{\langle \nabla U_f(x)\odot b, v\rangle^2}{||(\nabla U_f(x)\odot b)||^2} + 4\frac{\langle \nabla U_f(x)\odot b, v\rangle^2||(\nabla U_f(x)\odot b)||^2}{||(\nabla U_f(x)\odot b)||^4}\\ &= ||v||^2.\end{aligned}$$

- $R_f(x,b)R_f(x,b) = \mathbb{I}$:

$$\begin{aligned}R_f(x,b)R_f(x,b) &= \Big(\mathbb{I} - 2\frac{(\nabla U_f(x)\odot b)(\nabla U_f(x)\odot b)^T}{||(\nabla U_f(x)\odot b)||^2}\Big)\Big(\mathbb{I} - 2\frac{(\nabla U_f(x)\odot b)(\nabla U_f(x)\odot b)^T}{||(\nabla U_f(x)\odot b)||^2}\Big)\\ &= \mathbb{I} - 4\frac{(\nabla U_f(x)\odot b)(\nabla U_f(x)\odot b)^T}{||(\nabla U_f(x)\odot b)||^2}\\ &\quad + 4\frac{(\nabla U_f(x)\odot b)(\nabla U_f(x)\odot b)^T(\nabla U_f(x)\odot b)(\nabla U_f(x)\odot b)^T}{||(\nabla U_f(x)\odot b)||^4}\\ &= \mathbb{I} - 4\frac{(\nabla U_f(x)\odot b)(\nabla U_f(x)\odot b)^T}{||(\nabla U_f(x)\odot b)||^2} + 4\frac{(\nabla U_f(x)\odot b)(\nabla U_f(x)\odot b)^T}{||(\nabla U_f(x)\odot b)||^2}\\ &= \mathbb{I}.\end{aligned}$$

- $\langle \nabla_x U_f(x), b \odot R_f(x,b)v\rangle = -\langle \nabla_x U_f(x), b \odot v\rangle$:

$$\begin{aligned}\langle \nabla_x U_f(x), b \odot R_f(x,b)v\rangle &= \sum_i \left\{ \frac{\partial U_f(x)}{\partial x_i} b_i \left(v_i - 2\Big(\frac{\partial U_f(x)}{\partial x_i}\Big)\frac{\sum_j b_j \frac{\partial U_f(x)}{\partial x_j} v_j}{\sum_k b_k \Big(\frac{\partial U_f(x)}{\partial x_k}\Big)^2}\right)\right\}\\ &= \sum_i b_i \frac{\partial U_f(x)}{\partial x_i} v_i - 2\sum_i b_i \Big(\frac{\partial U_f(x)}{\partial x_i}\Big)^2 \frac{\sum_j b_j \frac{\partial U_f(x)}{\partial x_j} v_j}{\sum_k b_k \Big(\frac{\partial U_f(x)}{\partial x_k}\Big)^2}\\ &= -\sum_i b_i \frac{\partial U_f(x)}{\partial x_i} v_i\\ &= -\langle \nabla_x U_f(x), b \odot v\rangle.\end{aligned}$$

□

Proof of Lemma 3.1

Proof. Under the Masked BPS formulation, the state z and invariant distribution φ may be split into separate d-dimensional components $z = (x, v, b)$ and $\varphi(\mathrm{d}x, \mathrm{d}v, \mathrm{d}b) = \pi(\mathrm{d}x)\psi(\mathrm{d}v)\rho_{\mathcal{B}}(\mathrm{d}b)$. Let bounce events are indexed by $\imath \in \{1, \ldots, m\}$, and refresh event indexed at $i = 0$. In order to complete the proof, one must may verify the following:

1. There exists a φ-preserving mapping $\mathcal{S} : \mathcal{Z} \to \mathcal{Z}$, hence:

$$\varphi(\mathrm{d}z) = \varphi(\mathcal{S}(\mathrm{d}z)). \tag{15}$$

2. The event rates $\{\lambda_i\}_{i=0}^m$ satisfy:

$$\sum_{i=0}^{m}[\lambda_i(\mathcal{S}(z)) - \lambda_i(z)] = \nabla \cdot \phi(z) - \langle \nabla U(z), \phi(z)\rangle. \tag{16}$$

3. Each transition kernel Q_i for $i \in \{0, \ldots, m\}$ satisfy:

$$\int \lambda_i(z) Q_i(z, \mathrm{d}z')\varphi(\mathrm{d}z) = \varphi(\mathcal{S}^{-1}(\mathrm{d}z'))\lambda_i(\mathcal{S}(z')). \tag{17}$$

Consider the mapping $\mathcal{S}(x, v, b) = (x, -v, b)$. It is clear condition (15) holds given that $||v||_2^2 = ||-v||_2^2$, and ψ is the standard Gaussian distribution. Additionally, note that $\mathcal{S} = \mathcal{S}^{-1}$ (i.e. $\mathcal{S}$ is an involution), hence $\varphi(z) = \varphi(\mathcal{S}(z)) = \varphi(\mathcal{S}^{-1}(z))$ which will be used for the remaining conditions.

As $\phi(z) = (b \odot v, \mathbf{0}_d, \mathbf{0}_d)$, it is clear $\nabla \cdot \phi = 0$. λ_{sync} is constant, so:

$$\begin{aligned}
\sum_{i=0}^{m}[\lambda_i(\mathcal{S}(z) - \lambda_i(z)] &= \sum_{f\in F}[\lambda_f(x, b \odot -v) - \lambda_f(x, b \odot v)] && (18)\\
&= \sum_{f\in F}[\max\{0, -\langle \nabla_x U_f(x), b \odot v)\rangle\} - \max\{0, \langle \nabla_x U_f(x), b \odot v\rangle\}] && (19)\\
&= \sum_{f\in F} -\langle \nabla_x U_f(x), b \odot v\rangle && (20)\\
&= -\langle \nabla_x U(x), b \odot v\rangle && (21)
\end{aligned}$$

Hence, condition 2 is satisfied.

Condition 3 is trivial for the refresh/ synchronisation event:

$$\begin{aligned}
\int \lambda_0(z) Q_0(z, \mathrm{d}z')\varphi(\mathrm{d}z) &= \int \lambda_{sync}\delta_x(\mathrm{d}x')\psi(\mathrm{d}v')\rho_{\mathcal{B}}(\mathrm{d}b')\pi(\mathrm{d}x)\psi(\mathrm{d}v)\rho_{\mathcal{B}}(\mathrm{d}b) && (22)\\
&= \lambda_{sync}\psi(\mathrm{d}v')\rho_{\mathcal{B}}(\mathrm{d}b')\pi(\mathrm{d}x') && (23)\\
&= \lambda_{sync}\psi(-\mathrm{d}v')\rho_{\mathcal{B}}(\mathrm{d}b')\pi(\mathrm{d}x') && (24)\\
&= \lambda_0(\mathcal{S}(z'))\varphi(\mathcal{S}^{-1}(\mathrm{d}z')). && (25)
\end{aligned}$$

For the bounce events:

$$\begin{aligned}
\int \lambda_f(z) Q_f(z, \mathrm{d}z')\varphi(\mathrm{d}z) &= \int \lambda_f(x, b \odot v)\delta_x(x')\delta_b(b')\delta_{R_f(x,b)v}(v')\pi(\mathrm{d}x)\psi(\mathrm{d}v)\rho_{\mathcal{B}}(\mathrm{d}b) && (26)\\
&= \lambda_f(x', b' \odot R_f(x, b)v)\pi(\mathrm{d}x')\psi(\mathrm{d}v')\rho_{\mathcal{B}}(\mathrm{d}b') && (27)\\
&= \max\{0, \langle \nabla_x U(x'), b' \odot R_f(x, b)v\rangle\}\pi(\mathrm{d}x')\psi(\mathrm{d}v')\rho_{\mathcal{B}}(\mathrm{d}b') && (28)\\
&= \max\{0, \langle \nabla_x U(x'), b' \odot -v\rangle\}\pi(\mathrm{d}x')\psi(\mathrm{d}v')\rho_{\mathcal{B}}(\mathrm{d}b') && (29)\\
&= \lambda_f(\mathcal{S}(z'))\varphi(\mathcal{S}^{-1}(\mathrm{d}z')). && (30)
\end{aligned}$$

The penultimate step uses the identity $\langle \nabla_x U_f(x), b \odot R_f(x, b)v\rangle = -\langle \nabla_x U_f(x), b \odot v\rangle$ from Lemma 6.1. □

Indirect Proof of 3.1

Proof. Theorem 3.1 follows directly from Proposition 2 of Vanetti et al. 2017 given the conditions of this proposition are satisfied according to Lemma 3.1. □

Direct Proof of Theorem 3.1

Proof. To prove invariance, one may appeal to (Davis 1993, Theorem 24.7) and verify that the extended generator integrates to 0.

The integral of this extended generator is:

$$\int_b \int_v \int_x \mathcal{L}h(x,v,b)\pi(\mathrm{d}x)\psi(\mathrm{d}v)\rho_{\mathcal{B}}(\mathrm{d}b)$$

$$= \int_b \int_v \int_x \langle \nabla_x h(x,v,b), b \odot v \rangle \pi(\mathrm{d}x)\psi(\mathrm{d}v)\rho_{\mathcal{B}}(\mathrm{d}b) \tag{31}$$

$$+ \sum_{f\in F} \int_x \int_b \int_v \lambda_f(x, b\odot v)\{h(x, R_f(x,b)v, b) - h(x,v,b)\}\pi(\mathrm{d}x)\psi(\mathrm{d}v)\rho_{\mathcal{B}}(\mathrm{d}b) \tag{32}$$

$$+ \lambda_{sync} \int_x \int_b \int_v \int_{b',v'} \{h(x,v',b') - h(x,b,v)\}\psi(\mathrm{d}v')\rho_{\mathcal{B}}(\mathrm{d}b')\pi(\mathrm{d}x)\psi(\mathrm{d}v)\rho_{\mathcal{B}}(\mathrm{d}b). \tag{33}$$

Term (31)
Similar to the proof of the invariance of the Local BPS in Bouchard-Côté et al. 2018, it can be seen that term (31) may be rewritten as follows:

$$\int_x \int_b \int_v \langle \nabla_x h(x,v,b), b\odot v\rangle \pi(\mathrm{d}x)\psi(\mathrm{d}v)\rho_{\mathcal{B}}(\mathrm{d}b) = \int_x \int_b \int_v \langle \nabla_x U(x), b\odot v\rangle h(x,v,b)\pi(\mathrm{d}x)\psi(\mathrm{d}v)\rho_{\mathcal{B}}(\mathrm{d}b). \tag{34}$$

Re-writing the inner-product in term (34):

$$\int_x \int_b \int_v \langle \nabla_x h(x,v,b), b\odot v\rangle \pi(\mathrm{d}x)\psi(\mathrm{d}v)\rho_{\mathcal{B}}(\mathrm{d}b) = \int_b \int_v \int_x \left\{ \sum_i \frac{\partial h}{\partial x_i}(x,v,b) b_i v_i \right\} \pi(\mathrm{d}x)\psi(\mathrm{d}v)\rho_{\mathcal{B}}(\mathrm{d}b). \tag{35}$$

Now focusing on the inner integral terms, one may use integration by parts and the chain rule. Note that in an abuse of notation the RN density of each measure with respect to some dominating measure is denoted the same as the measure itself.

$$\int_{x_i} \frac{\partial h}{\partial x_i}(x,v,b) b_i v_i \pi(x) dx_i = h(x,v,b) b_i v_i \pi(x) - \int_{x_i} h(x,v,b) \frac{\partial \pi}{\partial x_i}(x) b_i v_i dx \tag{36}$$

$$= h(x,v,b) b_i v_i \pi(x) + \int_{x_i} h(x,v,b) \frac{\partial U}{\partial x_i}(x) b_i v_i \pi(x) dx_i. \tag{37}$$

Given again h is bounded and each v_i is centred at 0 under integration, the first term in equation 37 will reduce to 0 under integration with respect to ψ. By re-introducing the integrals with respect to ψ and $\rho_{\mathcal{B}}$ one can then see that equation (34) holds.

Term (32)
As ψ is the standard normal distribution, $\psi(\mathrm{d}v) = \psi(R_f(x,b)\mathrm{d}v)$ due to equation (12) in lemma 6.1. Using, this and also the equation (13) in lemma 6.1, a change of variables: $v \to R_f(x,b)v$ on the first term in equation (39) allows term (32) to be rearranged as:

$$\sum_{f\in F} \int_x \int_b \int_v \lambda_f(x, b\odot v)\{h(x, R_f(x,b)v, b) - h(x,v,b)\}\pi(\mathrm{d}x)\psi(\mathrm{d}v)\rho_{\mathcal{B}}(\mathrm{d}b) \tag{38}$$

$$= \sum_{f\in F} \int_x \int_b \int_v \lambda_f(x, b\odot v) h(x, R_f(x,b)v, b)\pi(\mathrm{d}x)\psi(\mathrm{d}v)\rho_{\mathcal{B}}(\mathrm{d}b)$$

$$- \int_x \int_b \int_v \lambda_f(x, b\odot v) h(x,v,b)\pi(\mathrm{d}x)\psi(\mathrm{d}v)\rho_{\mathcal{B}}(\mathrm{d}b) \tag{39}$$

$$= \sum_{f\in F} \int_x \int_b \int_v \lambda_f(x, b\odot R_f(x,b)v) h(x,v,b)\pi(\mathrm{d}x)\psi(\mathrm{d}v)\rho_{\mathcal{B}}(\mathrm{d}b)$$

$$- \int_x \int_b \int_v \lambda_f(x, b\odot v) h(x,v,b)\pi(\mathrm{d}x)\psi(\mathrm{d}v)\rho_{\mathcal{B}}(\mathrm{d}b) \tag{40}$$

$$= \sum_{f\in F} \int_x \int_b \int_v [\lambda_f(x, b\odot R_f(x,b)v) - \lambda_f(x, b\odot v)] h(x,v,b)\pi(\mathrm{d}x)\psi(\mathrm{d}v)\rho_{\mathcal{B}}(\mathrm{d}b). \tag{41}$$

Using $\lambda_f(x, b \odot v) = \max\{0, \langle \nabla_x U_f(x), b \odot v \rangle\}$ and equation (14) in lemma 6.1, term (32) may again be rewritten:

$$= \sum_{f \in F} \int_x \int_b \int_v [\max\{0, \langle \nabla_x U_f(x), b \odot R_f(x,b)v) \rangle\} - \max\{0, \langle \nabla_x U_f(x), b \odot v \rangle\}]$$
$$h(x, v, b)\pi(\mathrm{d}x)\psi(\mathrm{d}v)\rho_{\mathcal{B}}(\mathrm{d}b) \tag{42}$$
$$= \sum_{f \in F} \int_x \int_b \int_v [\max\{0, -\langle \nabla_x U_f(x), b \odot v \rangle\} - \max\{0, \langle \nabla_x U_f(x), b \odot v \rangle\}]$$
$$h(x, v, b)\pi(\mathrm{d}x)\psi(\mathrm{d}v)\rho_{\mathcal{B}}(\mathrm{d}b) \tag{43}$$
$$= -\sum_{f \in F} \int_x \int_b \int_v \langle \nabla_x U_f(x), b \odot v) \rangle h(x, v, b)\pi(\mathrm{d}x)\psi(\mathrm{d}v)\rho_{\mathcal{B}}(\mathrm{d}b) \tag{44}$$
$$= -\int_x \int_b \int_v \sum_{f \in F} \langle \nabla_x U_f(x), b \odot v) \rangle h(x, v, b)\pi(\mathrm{d}x)\psi(\mathrm{d}v)\rho_{\mathcal{B}}(\mathrm{d}b) \tag{45}$$
$$= -\int_x \int_b \int_v \langle \nabla_x U(x), b \odot v) \rangle h(x, v, b)\pi(\mathrm{d}x)\psi(\mathrm{d}v)\rho_{\mathcal{B}}(\mathrm{d}b). \tag{46}$$

It is clear that terms (31) and (32) are equivalent to (34) and (46) respectively, hence sum to 0.

It is not difficult to show term (33) is also 0, hence by (Davis 1993, Theorem 24.7) the procedure has the correct invariant distribution. □

Indirect Proof of Theorem 3.2

Proof. Theorem 3.2 follows directly from Proposition 3 of Vanetti et al. 2017 given the doubly stochastic conditions of this proposition are satisfied for each factorisation according to Lemma 3.1 and the process under dynamic factorisation admits the correct invariant distribution as detailed in Section 3.2. □

Direct Proof of Theorem 3.2

Proof. The proof follows a similar structure to that of Theorem 3.1, by using (Davis 1993, Theorem 24.7) under condition:

$$\int \int \int \int \mathcal{L}h(x, v, b, F)\pi(\mathrm{d}x)\psi(\mathrm{d}v)\rho_{\mathcal{B}_F}(\mathrm{d}b)\mu(\mathrm{d}F) = 0. \tag{47}$$

The generator of the Masked BPS with dynamic factorisation is given by Lemma 3.3:

$$\mathcal{L}h(x,v, b, F) =$$
$$\langle \nabla_x h(x, v, b, F), b \odot v \rangle \tag{48}$$
$$+ \sum_{f \in F} \lambda_f(x, b \odot v)\{h(x, R_f(x, b)v, b, F) - h(x, v, b, F)\} \tag{49}$$
$$+ \lambda_{sync} \int \{h(x, v', b', F') - h(x, v, b, F)\}\psi(\mathrm{d}v')\rho_{\mathcal{B}_F}(\mathrm{d}b')\mu_{\mathcal{F}}(\mathrm{d}F').$$

One can see that the first two terms of $\mathcal{L}h$ integrate to null using similar arguments as those in the proof of theorem 3.1:

$$\int \ldots \int \langle \nabla_x h(x, v, b, F), b \odot v \rangle + \sum_{f \in F} \lambda_f(x, b \odot v)\{h(x, R_f(x, b)v, b, F) - h(x, v, b, F)\}\pi(\mathrm{d}x)\psi(\mathrm{d}v)\rho_{\mathcal{B}_F}(\mathrm{d}b)\mu(\mathrm{d}F) = 0.$$

It is also clear that the final term in $\mathcal{L}h$ is also null under integration:

$$\int \ldots \int \lambda_{sync} \int \{h(x, v', b', F') - h(x, v, b, F)\}\psi(\mathrm{d}v')\rho_{\mathcal{B}_F}(\mathrm{d}b')\mu_{\mathcal{F}}(\mathrm{d}F')\pi(\mathrm{d}x)\psi(\mathrm{d}v)\rho_{\mathcal{B}_F}(\mathrm{d}b)\mu(\mathrm{d}F) = 0.$$

□

6.4 EXPERIMENT DETAILS

6.4.1 Chain Example

For the length $d = 7$ chain Gaussian Markov random field, one can see that the ESS of x_2, which has non-zero probability of being masked, increases at a lower rate than non-masked x_0 node. It is also clear that the Masked BPS is inferior to the Local BPS in such a setting, as the dimension is small.

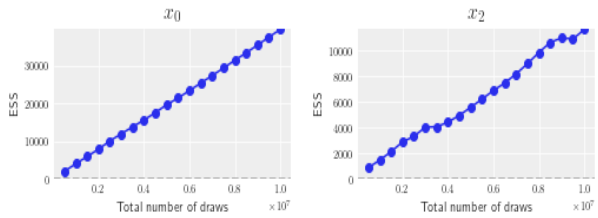

Figure 8: ESS evolution of x_0 and x_2 under the Masked BPS

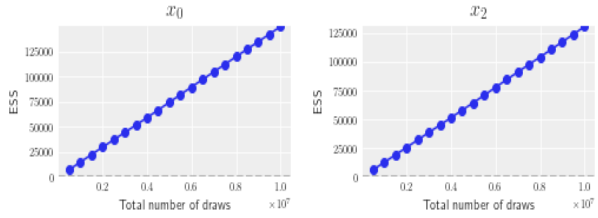

Figure 9: ESS evolution of x_0 and x_2 under the Local BPS

For the higher dimensional chain models, the pairwise precision was set to 0.5. The factorisation was given by blocks of adjacent variables, each of length 100 with overlap of length 1. Hence:

$$\pi(x_{0:99}) \prod_{i=1}^{n-1} \pi(x_{(99i+1):(99(i+1))} | x_{99i})$$

The mask-set consists of the vectors of 1's in all but $\min(\lfloor n/3 \rfloor, 29)$ entries. The coordinates selected to be masked are those in multiples of 99 joining adjacent factors, uniformly sampled. Refresh rate was set to $\lambda_{sync} = 0.01$.

6.4.2 Hierarchical Example

Recall the hierarchical Gaussian MRF may be factorised as:

$$\pi(x_0) \prod_{i=1}^{d} \pi(x_i | x_0)$$

Each factor consists of two parameters, the global parameter x_0 and local x_i $i > 0$, where the joint relationship is conditional Gaussian with pair-wise precision 0.5.

The mask-set for this example consists of two vectors. The first mask is formed by nulling the coordinate corresponding to the global parameter x_0 and leaving the remaining d local parameters unmasked. This permits parallel updates of all local unmasked parameters. The second vector consists of keeping all variables unmasked, coinciding with the Local BPS. The first and second mask vectors were selected with probabilities 0.8 and 0.2 respectively. The trivial mask vector of 1's was introduced to ensure ergodicity. Experiments using the global parameter and a subset of local parameters yielded poor mixing results for the global parameter when the number of active local parameters was small.

When the number of local parameters and hence sub-graphs exceeded the number of worker processes, multiple factor groups were assigned to the same worker-process.

A refresh rate of $\lambda_{sync} = 0.1$ was used.