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

James Thornton[1], George Deligiannidis[1], and Arnaud Doucet[1]

[1]Department of Statistics., University of Oxford

February 23, 2021

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. We extend the basic procedure by introducing a dynamic factorization scheme of the target distribution to reduce boundary effects commonly associated to fixed partitioning. We establish the validity of the proposed methods theoretically 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

MCMC procedures are a well understood and widely applied class of algorithms to perform statistical inference for complex models [Robert and Casella, 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 parallelize, 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 and de With, 2012, Bouchard-Côté et al., 2018, Zhao and Bouchard-Côté, 2019], Zig-Zag [Bierkens et al., 2019], Stochastic BPS [Pakman et al., 2017] and Coordinate Sampler [Wu and Robert, 2020] are rejection-free and benefit from forms of sub-sampling. In addition, as with Hamiltonian Monte Carlo (HMC) [Duane et al., 1987, Neal, 2011] and Langevin Monte Carlo,

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. 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 [Daskalakis et al., 2018, Gonzalez et al., 2011, Jacob et al., 2020, Song and Moore, 2017, Terenin et al., 2020], 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 Contribution

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 and Moore [2017].

1.2 Setting and Notation

We provide methodology for the task of sampling from a target probability distribution 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 factorized as in Equation (1), though this factorization 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 factorization:

$$\pi(x) \propto \gamma(x) \qquad\qquad \gamma(x) = \prod_{f \in F} \gamma_f(x_f), \tag{1}$$

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 factorized form:

$$U(x) = \sum_{f \in F} U_f(x_f) \qquad\qquad U_f(x_f) = -\log \gamma_f(x_f). \tag{2}$$

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 parameterizes 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 and de With, 2012, Bouchard-Côté et al., 2018] is one of the most well-studied theoretically [Deligiannidis et al., 2019, Durmus et al., 2020] with various extensions such as the Local

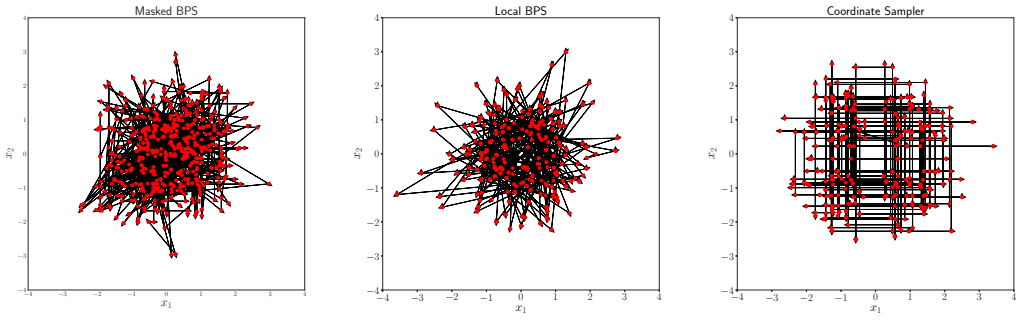


Figure 1: Path trajectory of Masked BPS, Local BPS and Coordinate Sampler for 500 events targeting an Isotropic Gaussian.

BPS [Bouchard-Côté et al., 2018], Stochastic BPS Pakman et al. [2017], and Generalized BPS [Wu and Robert, 2017]. The Masked BPS proposed here is another variant of the BPS, taking inspiration from the Local BPS and Coordinate Sampler [Wu and Robert, 2020].

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 subsampling. 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.

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 $\tau = \min\{\tau_{ref}, \{\tau_f\}_f\}$.

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**
   $\frac{\mathrm{d}x_t}{\mathrm{d}t} = v_t \qquad \frac{\mathrm{d}v_t}{\mathrm{d}t} = 0$

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

$$R_f(x)v = \Big(\mathbb{I} - 2 \frac{\nabla U_f(x) \big(\nabla U_f(x) \big)^T}{||\nabla U_f(x)||^2} \Big) v.$$

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 $\mathcal{V} = \{\pm e_i | i \in \{1, \ldots, d\}\}$ and $\{e_i\}_{i=1}^d$ form the standard basis for $\mathbb{R}^d$. 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 $\lambda(x, \mp e_i)$.

1. **Event Rate**
   $\lambda(x, v) = \lambda_{ref} + \max\{0, \langle \nabla U(x), v \rangle\}$
2. **Deterministic State Propagation**
   $\frac{\mathrm{d}x_t}{\mathrm{d}t} = v_t \qquad \frac{\mathrm{d}v_t}{\mathrm{d}t} = 0$
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 generalizing the Coordinate Sampler's deterministic dynamics to 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 $N \in \{1, \ldots, d\}$. 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 generalized to blocks rather than coordinates. 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.

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 $z_t = (x_t, v_t, b_t) \in \mathbb{R}^d \times \mathbb{R}^d \times \mathcal{B}$ and may be implemented using Algorithm 3.

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**
   $\frac{\mathrm{d}x_t}{\mathrm{d}t} = b_t \odot v_t \qquad \frac{\mathrm{d}v_t}{\mathrm{d}t} = 0 \qquad \frac{\mathrm{d}b_t}{\mathrm{d}t} = 0$
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 $v \to R_f(x, b)v$, given by:

$$R_f(x, b) = \Big(\mathbb{I} - 2\frac{(\nabla U_f(x) \odot b)(\nabla U_f(x) \odot b)^T}{||(\nabla U_f(x) \odot b)||^2}\Big). \tag{3}$$

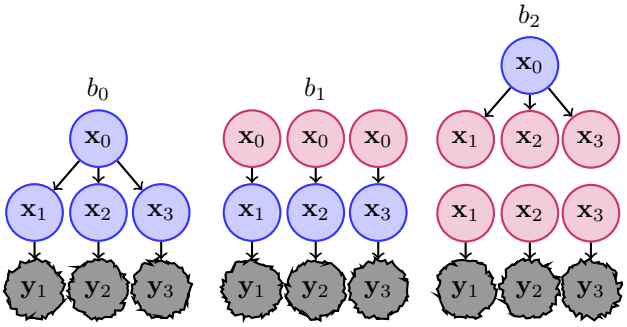


Figure 2: Blue indicates non-masked parameters, red indicates masked parameters, and grey indicates constants i.e. data when sampling a posterior. Masks: $b_0 = (1, 1, 1, 1)$, $b_1 = (0, 1, 1, 1)$, $b_2 = (1, 0, 0, 0)$

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 sub-sampling, 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 synchronization 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. Figure 2 depicts a simple hierarchical model and sub-graphs under masks b_0, b_1, b_2.

Algorithm 1: 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

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;

Lemma 3.1. *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 3 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 A.2 in the Appendices.

A refresh event for auxiliary variables b and v at synchronization 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. During refresh events, the variables b, v and τ_{sync} are sampled independently of the state of the process. This means that for

Algorithm 3: 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 / Synchronization 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

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 initialization. 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 Factorization

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 factorization, one may exploit the doubly stochastic PDMP formulation detailed in Pakman et al. [2017] and Vanetti et al. [2017] to allow for a dynamic factorization of the Local and Masked BPS.

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

$$\pi(x, F) = \pi(x|F)\mu_{\mathcal{F}}(\{F\}) = \mu_{\mathcal{F}}(\{F\}) \prod_{f \in F} \pi_f(x_f)$$

It is straightforward to show that the factorization may be marginalized to recover the target distribution.

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

In order to preserve the efficiency gains from sub-sampling in the Local BPS and Masked BPS, it is convenient to perform the factorization transitions at refresh or synchronization events. Indeed, given that the choice of mask-set $\mathcal{B}$ is dependent on factorization F, one may choose to sample both F and $\mathcal{B}$ jointly at synchronization events. This leads to an algorithm very similar to Algorithm 3, with the addition of sampling F at synchronization times.

Consider a chain shaped Gaussian Markov Random Field (MRF) of length $d = 7$ where each variable is standard normal, with pairwise precision $\rho = 0.5$, as depicted in Figure 3.

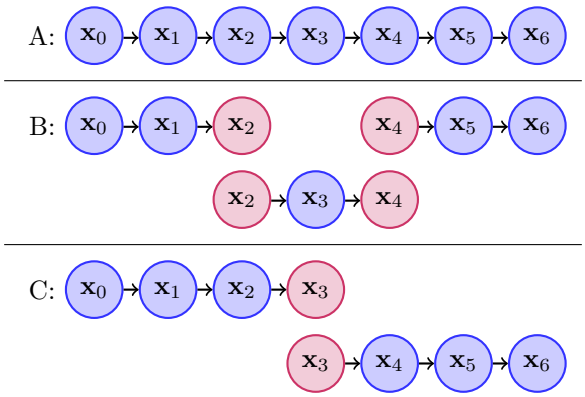


Figure 3: Diagram of dimension $d = 7$ Gaussian chain model, with example separations into subgraphs under different factorization schemes.

One may split the chain into three factors using factorization B in Figure 3 and masking x_2 and x_4. Masking x_3 with factorization C results in a different split. If there is strong inter-dependency between x_2 and x_1 (or x_4 and x_5), then mixing will be slow under factorization B as x_2 (and/or x_4) will be static for periods. However, mixing may be faster under factorization C. If the dependency is not known a priori, then one may use a dynamic factorization to reduce the risk of choosing an inefficient factorization. Indeed one may view choosing the factorization, or collection of factorizations, in the same way as tuning a hyper-parameter to improve algorithm performance. For the above described model, the Masked BPS with dynamic scheme using factorizations B and C with equal probability delivers effective sample size (ESS) per second of 665.6. The static Masked BPS under factorization B has ESS/s of 579.2. This demonstrated the benefit of the dynamic extension.

Theorem 3.2. *The Masked BPS with Dynamic Factorization 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 A.2 to verify conditions of Proposition 3 in Vanetti et al. [2017] for doubly stochastic PDMPs, as shown in the Appendices.

3.3 Discussion

3.3.1 Related Samplers

The mask-set $\mathcal{B}$, factorization-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 factorization 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 factorization-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 factorization set to include both the trivial single factor factorization in addition to a non-trivial factorization, 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 factorization set of a single factor, one recovers a Coordinate Sampler style procedure. In such a setting, the velocity component is updated according to $v \odot e_i \to -v \odot e_i$ under operator $R_f(x, e_i)$ between refresh events. The primary difference to the Coordinate Sampler is that 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 factorization set.

3.3.2 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. 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. This makes the parallel computing capabilities of the Masked BPS 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 factorization may hedge against a poor choice of factorization, an unfortunate consequence of having multiple factorizations is that one must be able to sample bounce event proposal times for each factor in each factorization. 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.

4 Numerical Experiments

The dynamic version of the Masked BPS, referred to as Masked BPS henceforth, is compared to Local BPS and HMC in numerous experiments on synthetic and real data.

The HMC NUTS implementation from PyMC3 [Salvatier et al., 2016] is used for benchmarking and is used with the NumPy backend for fair comparison to the Masked BPS and Local BPS implementation, which also use NumPy [Harris et al., 2020]. Distributed computing library Ray [Moritz et al., 2018] is used to introduce multi-process parallelism for the Masked BPS. Similar computational resources are used for each of the methods due to multi-threading for the Local BPS and HMC. The Local BPS and Masked BPS are used with refresh rates $\lambda_{sync} = \lambda_{ref} = 0.01$ and average effective sample size (ESS) per second is presented over 25 runs. Up to 12 cores are used for the Masked BPS.

Further technical implementation details such as dynamic factorization and mask schemes as well as results tables and code are given in the supplementary material.

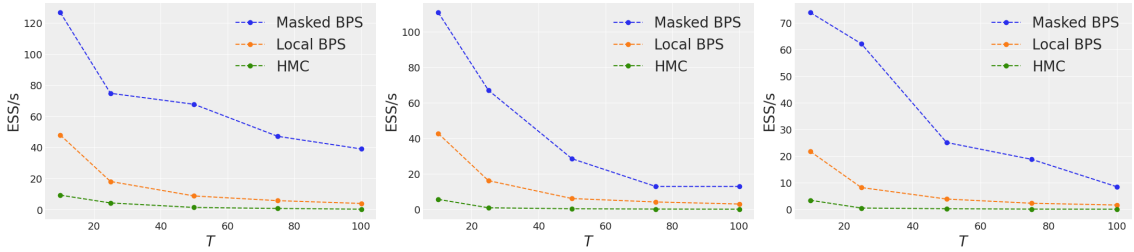
4.1 Gaussian State Space Model

Consider latent variables $x_t \in \mathbb{R}^d$ for $t \in \{1, \dots, T\}$ of a simple Gaussian state-space model where $x_0 \sim \mathcal{N}(\mu_0, \Sigma_0)$ and for $t \geq 1$

$$\begin{aligned} x_t &= Fx_{t-1} + u_t, & u_t &\sim \mathcal{N}(0, \Sigma_U), \\ y_t &= x_t + v_t, & v_t &\sim \mathcal{N}(0, \Sigma_V). \end{aligned}$$

The Masked BPS was compared to both HMC (NUTS) and the Local BPS for a variety of d, T configurations. In each experiment $F = 0.5\mathbb{I}_d$, $\mu_0 = 0$ and $\Sigma_0 = \Sigma_U = \Sigma_V = \mathbb{I}_d$ where $\mathbb{I}_d$ is the d dimensional identity matrix. In the Local BPS, the graph was factored per time-step. As all factors are Gaussian, event times were computed using the time-scale transformation of Bouchard-Côté et al. [2018, Section 2.3]. Dynamic factorization was performed by splitting and grouping the model temporally across $\min(T/5, 12)$ processes.

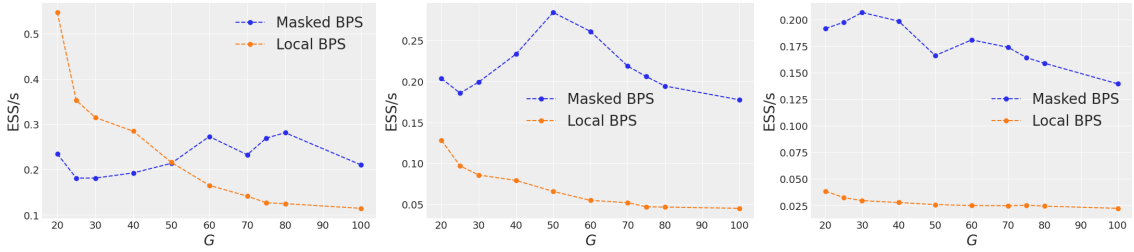
Figure 4 illustrates the superior performance of the Masked BPS for a range of size models compared to the other methods considered.



(a) $d = 10$ (b) $d = 25$ (c) $d = 50$

Figure 4: Gaussian State Space Model ESS/s

4.2 Hierarchical Bayesian Logistic Regression



(a) $d = 10$ (b) $d = 25$ (c) $d = 50$

Figure 5: Hierarchical Bayesian Logistic Regression, ESS/s

The target distribution in a Bayesian logistic model is of the regression coefficients whereby $x_0 \in \mathbb{R}^d$ denotes global coefficients and $x_g \in \mathbb{R}^d$ denotes local coefficients for group $g \in \{1, \ldots, G\}$, Each group g has corresponding covariates $\xi_g \in \mathbb{R}^{N \times d}$ and $N = 10d$ observations, $y_g \in \{0, 1\}$, where

$$x_0 \sim \mathcal{N}(0, \Sigma_U), \qquad x_g = x_0 + v_t, \qquad v_t \sim \mathcal{N}(0, \Sigma_v),$$
$$p_g = \text{Logistic}(\xi_g^T x_g), \qquad y_g \sim \text{Bernoulli}(p_g).$$

The Masked BPS is compared to Local BPS for a range of d and G. Only a single factorization was used, and the graph was split my masking x_0 periodically. Event proposals for the logistic terms are sampled using alias tables, described in Bouchard-Côté et al. [2018, Appendix C]. It can be seen in Figure 5 that for the smallest models, $d = 10, G \in \{25, 50\}$ the Local BPS performs well, however the benefits of parallelization are realized for larger models, and the Masked BPS exhibits superior performance for all other configurations. For the Masked BPS, ESS/s initially increases with G as additional parallel computing resources are deployed. However, the number of parallel processes is limited to 12, hence multiple independent sub-graphs are assigned to the same worker process leading to lower ESS/s for high G. The HMC implementation used suffers instability for larger groups, G and is omitted here. One of the reasons the Masked BPS performs well here is due to the recycling of event time proposals. There tends to be high correlation between global variables x_0, and local variables x_g, this leads to slow exploration of the sample space. In the context of the Local BPS, any event involving x_0 triggers recalculation of event times for the other G factors involving x_0. This is expensive. If x_0 was masked, then the x_g can be updated independently without triggering recalculation of new event proposals.

4.3 Dynamic Bradley-Terry Model

In the simplest setting of the Bradley-Terry (BT) model, there are d players whereby player i has an associated skill quantity $x_i \in \mathbb{R}^d$. In a competition between player i and player j, the probability of player i winning is $\frac{e^{x_i}}{e^{x_i}+e^{x_j}}$. This is a difficult problem due to the sparsity of observations, as observed competition results correspond only to pairs of the unknown skill quantities. The values $\{x_i\}_i$ may be used for ranking players and predicting future results and has application in sports and video games. The dynamic extension to this model incorporates temporal dependency across the skill quantities, $(x^{(t)})_{t=1}^T$:

$$x_0 \sim \mathcal{N}(\mu_0, \Sigma_0), \qquad x^{(t)} = x^{(t-1)} + u_t, \quad u_t \sim \mathcal{N}(0, \Sigma_U),$$
$$y_{i,j}^{(t)} = \text{Bernoulli}\left(\frac{e^{x_i^{(t)}}}{e^{x_i^{(t)}} + e^{x_j^{(t)}}}\right).$$

Football match data has been taken from Curley and Malmedal [2014]:

England Soccer matches excluding draws, across English divisions from seasons $1950 - 2019$, $T = 69$, including only the $d = 61$ teams present in every season.

France, Germany Matches excluding draws, across all division from $1980 - 2019$, $T = 39$, including the $d = 99$ teams present in any season, for each country.

Belgium Soccer matches, excluding draws, from $1995 - 2019$, $T = 25$, including any of the $d = 36$ team that played.

The Masked BPS has been compared to the Local BPS for $\mu_0 = 0$, $\Sigma_0 = \Sigma_U = \mathbb{I}_d$. The Local BPS performs marginally better for the Belgium dataset due to the smaller time-series and fewer matches, leading to less relative benefit from splitting the factor graph. Though note these experiments fix the number of graph partitions to $\min(T/5, 12)$, setting a single partition would be the same as Local BPS. For larger models however, the Masked BPS shows significant outperformance.

Table 1: Dynamic BT model, ESS/s $\times 10^{-2}$

	d	T	Local BPS	Masked BPS
England	61	69	1.36	**8.11**
France	99	39	1.40	**8.74**
Germany	99	39	1.22	**11.31**
Belgium	36	25	**13.97**	12.78

5 Conclusion and Extensions

This work introduces the Masked Bouncy Particle Sampler and dynamic factorization extension. Many probabilistic models may be expressed as sparse factor graphs, the Masked BPS exploits this sparsity by splitting the model's factor graph into sub-graphs and performing piece-wise deterministic sampling on each sub-graph, in parallel. It has been shown that the Masked BPS outperforms state-of-the-art sampling procedures for many large sparse graphical models.

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 synchronizing worker processes. Recent developments by Terenin et al. [2020], Daskalakis et al. [2018], Feng et al. [2021] 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 PDMP setting.

References

Joris Bierkens, Paul Fearnhead, and Gareth Roberts. The zig-zag process and super-efficient sampling for Bayesian analysis of big data. *The Annals of Statistics*, 47(3):1288–1320, 2019.

Alexandre Bouchard-Côté, Sebastian J Vollmer, and Arnaud Doucet. The bouncy particle sampler: A nonreversible rejection-free Markov chain Monte Carlo method. *Journal of the American Statistical Association*, 113(522):855–867, 2018.

James Curley and Håkon Malmedal. engsoccerdata: Release v0.1.2, December 2014. URL `https://doi.org/10.5281/zenodo.13158`.

Constantinos Daskalakis, Nishanth Dikkala, and Siddhartha Jayanti. HOGWILD!-Gibbs can be panaccurate. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 32–41, 2018.

Mark HA Davis. *Markov Models & Optimization*. Routledge, 1993.

George Deligiannidis, Alexandre Bouchard-Côté, and Arnaud Doucet. Exponential ergodicity of the bouncy particle sampler. *The Annals of Statistics*, 47(3):1268–1287, 2019.

Simon Duane, Anthony D Kennedy, Brian J Pendleton, and Duncan Roweth. Hybrid Monte Carlo. *Physics Letters B*, 195(2):216–222, 1987.

Alain Durmus, Arnaud Guillin, and Pierre Monmarché. Geometric ergodicity of the bouncy particle sampler. *Annals of Applied Probability*, 30(5):2069–2098, 2020.

Paul Fearnhead, Joris Bierkens, Murray Pollock, and Gareth O Roberts. Piecewise deterministic Markov processes for continuous-time Monte Carlo. *Statistical Science*, 33(3):386–412, 2018.

Weiming Feng, Thomas P Hayes, and Yitong Yin. Distributed Metropolis sampler with optimal parallelism. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2121–2140. SIAM, 2021.

Joseph Gonzalez, Yucheng Low, Arthur Gretton, and Carlos Guestrin. Parallel Gibbs sampling: From colored fields to thin junction trees. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 324–332, 2011.

Charles R. Harris, K. Jarrod Millman, St'efan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. 2020.

Pierre E Jacob, John O'Leary, and Yves F Atchadé. Unbiased Markov chain Monte Carlo methods with couplings. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 82(3): 543–600, 2020.

Philipp Moritz, Robert Nishihara, Stephanie Wang, Alexey Tumanov, Richard Liaw, Eric Liang, Melih Elibol, Zongheng Yang, William Paul, Michael I. Jordan, and et al. Ray: A distributed framework for emerging ai applications. In *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*, OSDI'18, page 561–577, USA, 2018. USENIX Association. ISBN 9781931971478.

Radford M Neal. MCMC using Hamiltonian dynamics. *Handbook of Markov chain Monte Carlo*, 2 (11):2, 2011.

Ari Pakman, Dar Gilboa, David Carlson, and Liam Paninski. Stochastic bouncy particle sampler. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2741–2750. JMLR. org, 2017.

Elias AJF Peters and G. de With. Rejection-free Monte Carlo sampling for general potentials. *Physical Review E*, 85(2):026703, 2012.

Christian Robert and George Casella. *Monte Carlo Statistical Methods*. Springer Science & Business Media, 2013.

John Salvatier, Thomas V. Wiecki, and Christopher Fonnesbeck. Probabilistic programming in python using PyMC3. *PeerJ Computer Science*, 2:e55, apr 2016. doi: 10.7717/peerj-cs.55. URL `https://doi.org/10.7717/peerj-cs.55`.

Jun Song and David Moore. Parallel chromatic MCMC with spatial partitioning. In *Workshops at the Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

Alexander Terenin, Daniel Simpson, and David Draper. Asynchronous Gibbs sampling. In *International Conference on Artificial Intelligence and Statistics*, pages 144–154. PMLR, 2020.

Paul Vanetti, Alexandre Bouchard-Côté, George Deligiannidis, and Arnaud Doucet. Piecewise-deterministic Markov chain Monte Carlo. *arXiv preprint arXiv:1707.05296*, 2017.

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

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

Changye Wu and Christian P Robert. The coordinate sampler: A non-reversible Gibbs-like MCMC sampler. *Statistics and Computing*, 30:721–730, 2020.

Tingting Zhao and Alexandre Bouchard-Côté. Analysis of high-dimensional continuous time Markov chains using the local bouncy particle sampler. *arXiv preprint arXiv:1905.13120*, 2019.

A Invariance Proofs

Lemma A.1. *The bounce operation, $R_f(x,b)$, on the masked process enjoys the following properties:*

$$||R_f(x,b)v|| = ||v||, \tag{4}$$
$$R_f(x,b)R_f(x,b) = \mathbb{I}, \tag{5}$$
$$\langle \nabla_x U_f(x), b \odot R_f(x,b)v\rangle = -\langle \nabla_x U_f(x), b \odot v\rangle. \tag{6}$$

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

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

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

□

For ease of notation, let refresh events correspond to index 0 and bounce events be indexed by $i \in [m] = \{1, \ldots, m\}$ 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 A.2. *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 $i \in \{0, 1, \ldots, m\}$:*

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

Proof of Lemma A.2

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 be indexed by $i \in \{1, \ldots, m\}$, 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:

$$\varphi(\mathrm{d}z) = \varphi(\mathcal{S}(\mathrm{d}z)). \tag{7}$$

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

$$\sum_{i=0}^{m} [\lambda_i(\mathcal{S}(z)) - \lambda_i(z)] = \nabla \cdot \phi(z) - \langle \nabla U(z), \phi(z) \rangle. \tag{8}$$

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

$$\int \lambda_i(z) Q_i(z, \mathrm{d}z') \varphi(\mathrm{d}z) = \varphi(\mathcal{S}^{-1}(\mathrm{d}z')) \lambda_i(\mathcal{S}(z')). \tag{9}$$

Consider the mapping $\mathcal{S}(x, v, b) = (x, -v, b)$. It is clear condition (7) 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:

$$\sum_{i=0}^{m}[\lambda_i(\mathcal{S}(z) - \lambda_i(z)] = \sum_{f \in F}[\lambda_f(x, b \odot -v) - \lambda_f(x, b \odot v)] \tag{10}$$

$$= \sum_{f \in F}[\max\{0, -\langle \nabla_x U_f(x), b \odot v)\rangle\} - \max\{0, \langle \nabla_x U_f(x), b \odot v\rangle\}] \tag{11}$$

$$= \sum_{f \in F} -\langle \nabla_x U_f(x), b \odot v\rangle \tag{12}$$

$$= -\langle \nabla_x U(x), b \odot v\rangle \tag{13}$$

Hence, condition 2 is satisfied.

Condition 3 is trivial for the refresh/ synchronization event:

$$\int \lambda_0(z) Q_0(z, \mathrm{d}z')\varphi(\mathrm{d}z) = \int \lambda_{sync}\delta_x(\mathrm{d}x')\psi(\mathrm{d}v')\rho_{\mathcal{B}}(\mathrm{d}b')\pi(\mathrm{d}x)\psi(\mathrm{d}v)\rho_{\mathcal{B}}(\mathrm{d}b) \tag{14}$$

$$= \lambda_{sync}\psi(\mathrm{d}v')\rho_{\mathcal{B}}(\mathrm{d}b')\pi(\mathrm{d}x') \tag{15}$$

$$= \lambda_{sync}\psi(-\mathrm{d}v')\rho_{\mathcal{B}}(\mathrm{d}b')\pi(\mathrm{d}x') \tag{16}$$

$$= \lambda_0(\mathcal{S}(z'))\varphi(\mathcal{S}^{-1}(\mathrm{d}z')). \tag{17}$$

For the bounce events:

$$\int \lambda_f(z) Q_f(z, \mathrm{d}z')\varphi(\mathrm{d}z) = \int \lambda_f(x, b \odot v)\delta_x(x')\delta_b(b')\delta_{R_f(x,b)v}(v')\pi(\mathrm{d}x)\psi(\mathrm{d}v)\rho_{\mathcal{B}}(\mathrm{d}b) \tag{18}$$

$$= \lambda_f(x', b' \odot R_f(x, b)v)\pi(\mathrm{d}x')\psi(\mathrm{d}v')\rho_{\mathcal{B}}(\mathrm{d}b') \tag{19}$$

$$= \max\{0, \langle \nabla_x U(x'), b' \odot R_f(x, b)v\rangle\}\pi(\mathrm{d}x')\psi(\mathrm{d}v')\rho_{\mathcal{B}}(\mathrm{d}b') \tag{20}$$

$$= \max\{0, \langle \nabla_x U(x'), b' \odot -v\rangle\}\pi(\mathrm{d}x')\psi(\mathrm{d}v')\rho_{\mathcal{B}}(\mathrm{d}b') \tag{21}$$

$$= \lambda_f(\mathcal{S}(z'))\varphi(\mathcal{S}^{-1}(\mathrm{d}z')). \tag{22}$$

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 A.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 A.2. □

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:

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

$$= \int_b \int_v \int_x \langle \nabla_x h(x,v,b), b \odot v \rangle \pi(\mathrm{d}x)\psi(\mathrm{d}v)\rho_{\mathcal{B}}(\mathrm{d}b) \tag{23}$$

$$+ \sum_{f \in F} \int_x \int_b \int_v \lambda_f(x, b \odot v)\{h(x, R_f(x,b)v, b) - h(x,v,b)\}\pi(\mathrm{d}x)\psi(\mathrm{d}v)\rho_{\mathcal{B}}(\mathrm{d}b) \tag{24}$$

$$+ \lambda_{sync} \int_x \int_b \int_v \int_{b',v'} \{h(x,v',b') - h(x,b,v)\}\psi(\mathrm{d}v')\rho_{\mathcal{B}}(\mathrm{d}b')\pi(\mathrm{d}x)\psi(\mathrm{d}v)\rho_{\mathcal{B}}(\mathrm{d}b). \tag{25}$$

Term (23)
Similar to the proof of the invariance of the Local BPS in Bouchard-Côté et al. [2018], it can be seen that term (23) may be rewritten as follows:

$$\int_x \int_b \int_v \langle \nabla_x h(x,v,b), b \odot v \rangle \pi(\mathrm{d}x)\psi(\mathrm{d}v)\rho_{\mathcal{B}}(\mathrm{d}b) = \int_x \int_b \int_v \langle \nabla_x U(x), b \odot v \rangle h(x,v,b)\pi(\mathrm{d}x)\psi(\mathrm{d}v)\rho_{\mathcal{B}}(\mathrm{d}b). \tag{26}$$

Re-writing the inner-product in term (26):

$$\int_x \int_b \int_v \langle \nabla_x h(x,v,b), b \odot v \rangle \pi(\mathrm{d}x)\psi(\mathrm{d}v)\rho_{\mathcal{B}}(\mathrm{d}b) = \int_b \int_v \int_x \left\{ \sum_i \frac{\partial h}{\partial x_i}(x,v,b) b_i v_i \right\} \pi(\mathrm{d}x)\psi(\mathrm{d}v)\rho_{\mathcal{B}}(\mathrm{d}b). \tag{27}$$

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.

$$\int_{x_i} \frac{\partial h}{\partial x_i}(x,v,b) b_i v_i \pi(x) dx_i = h(x,v,b) b_i v_i \pi(x) - \int_{x_i} h(x,v,b) \frac{\partial \pi}{\partial x_i}(x) b_i v_i dx \tag{28}$$

$$= h(x,v,b) b_i v_i \pi(x) + \int_{x_i} h(x,v,b) \frac{\partial U}{\partial x_i}(x) b_i v_i \pi(x) dx_i. \tag{29}$$

Given again h is bounded and each v_i is centred at 0 under integration, the first term in equation 29 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 (26) holds.

Term (24)
As ψ is the standard normal distribution, $\psi(\mathrm{d}v) = \psi(R_f(x,b)\mathrm{d}v)$ due to equation (4) in lemma A.1. Using, this and also the equation (5) in lemma A.1, a change of variables: $v \to R_f(x,b)v$ on the first

term in equation (31) allows term (24) to be rearranged as:

$$\sum_{f\in F}\int_x\int_b\int_v \lambda_f(x,b\odot v)\{h(x,R_f(x,b)v,b)-h(x,v,b)\}\pi(\mathrm{d}x)\psi(\mathrm{d}v)\rho_{\mathcal{B}}(\mathrm{d}b) \tag{30}$$

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

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

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

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

$$=\sum_{f\in F}\int_x\int_b\int_v [\lambda_f(x,b\odot R_f(x,b)v)-\lambda_f(x,b\odot v)]h(x,v,b)\pi(\mathrm{d}x)\psi(\mathrm{d}v)\rho_{\mathcal{B}}(\mathrm{d}b). \tag{33}$$

Using $\lambda_f(x,b\odot v)=\max\{0,\langle\nabla_x U_f(x),b\odot v\rangle\}$ and equation (6) in lemma A.1, term (24) may again be rewritten:

$$=\sum_{f\in F}\int_x\int_b\int_v [\max\{0,\langle\nabla_x U_f(x),b\odot R_f(x,b)v)\rangle\}-\max\{0,\langle\nabla_x U_f(x),b\odot v\rangle\}]$$

$$h(x,v,b)\pi(\mathrm{d}x)\psi(\mathrm{d}v)\rho_{\mathcal{B}}(\mathrm{d}b) \tag{34}$$

$$=\sum_{f\in F}\int_x\int_b\int_v [\max\{0,-\langle\nabla_x U_f(x),b\odot v)\rangle\}-\max\{0,\langle\nabla_x U_f(x),b\odot v\rangle\}]$$

$$h(x,v,b)\pi(\mathrm{d}x)\psi(\mathrm{d}v)\rho_{\mathcal{B}}(\mathrm{d}b) \tag{35}$$

$$=-\sum_{f\in F}\int_x\int_b\int_v \langle\nabla_x U_f(x),b\odot v)\rangle h(x,v,b)\pi(\mathrm{d}x)\psi(\mathrm{d}v)\rho_{\mathcal{B}}(\mathrm{d}b) \tag{36}$$

$$=-\int_x\int_b\int_v \sum_{f\in F}\langle\nabla_x U_f(x),b\odot v)\rangle h(x,v,b)\pi(\mathrm{d}x)\psi(\mathrm{d}v)\rho_{\mathcal{B}}(\mathrm{d}b) \tag{37}$$

$$=-\int_x\int_b\int_v \langle\nabla_x U(x),b\odot v)\rangle h(x,v,b)\pi(\mathrm{d}x)\psi(\mathrm{d}v)\rho_{\mathcal{B}}(\mathrm{d}b). \tag{38}$$

It is clear that terms (23) and (24) are equivalent to (26) and (38) respectively, hence sum to 0.

It is not difficult to show term (25) 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 factorization according to Lemma A.2 and the process under dynamic factorization admits the correct invariant distribution as detailed in Section 3.2. □

Lemma A.3. *The generator $\mathcal{L}$ for the Masked BPS with dynamic factorization is given by:*

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

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:

$$\int\int\int\int \mathcal{L}h(x,v,b,F)\pi(\mathrm{d}x)\psi(\mathrm{d}v)\rho_{\mathcal{B}_F}(\mathrm{d}b)\mu(\mathrm{d}F) = 0. \tag{39}$$

The generator of the Masked BPS with dynamic factorization is given by Lemma A.3:

$$\begin{aligned}\mathcal{L}h(x,&v,b,F) = \\ &\langle \nabla_x h(x,v,b,F), b \odot v \rangle &(40)\\ &+ \sum_{f \in F} \lambda_f(x, b \odot v)\{h(x, R_f(x,b)v, b, F) - h(x,v,b,F)\} &(41)\\ &+ \lambda_{sync} \int \{h(x,v',b',F') - h(x,v,b,F)\}\psi(\mathrm{d}v')\rho_{\mathcal{B}_F}(\mathrm{d}b')\mu_{\mathcal{F}}(\mathrm{d}F').\end{aligned}$$

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:

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

□

B Additional Experiment Detail

All experiments were ran on a Google Cloud Platform (GCP) N1 Series 18vCPUs, 100GB RAM machine using Intel Broadwell CPU platform.

B.1 Gaussian State Space Model

The model consists of multiple Gaussian factors. Event proposal times for each factor may be simulated using time-scale transformation from Bouchard-Côté et al. [2018]. Dynamic masking and factorization was achieved by using the most granular factorization of masking up $S = min(12, T/5)$ d-dimensional x_t nodes along the t index then grouping factors sequentially into subgraphs. The Local BPS used the most granular factorization without grouping. The number of masked variables was chosen to balance the benefits of extra computational resources with the decreased efficiency of sampling.

Table 2: Gaussian State Space Model, ESS/s

d	Method	$T = 10$	25	50	100
10	Local BPS	47.89	18.02	8.71	3.93
	Masked BPS	**126.75**	**74.77**	**67.68**	**41.30**
	HMC	9.21	4.21	1.30	0.20
25	Local BPS	34.63	13.94	6.12	3.03
	Masked BPS	**152.18**	**43.64**	**28.9**	**12.94**
	HMC	5.66	0.92	0.38	0.13
50	Local BPS	21.76	8.19	3.87	1.62
	Masked BPS	**73.91**	**62.17**	**25.11**	**8.47**
	HMC	3.42	0.49	0.22	0.06

B.2 Hierarchical Bayesian Logistic Regression

This model consists of Gaussian factors ($\pi(x_0)$, $\pi(x_g|x_o)$) and logistic factors $\prod_{i=1}^{N} \pi(y_i|x_g)$. Event time proposals for Gaussian factors were simulated by time-scale transformation and the proposals for logistic terms were sampled using thinning and the alias method, described in detail in [Bouchard-Côté et al., 2018, Appendix C.1].

The static factorization of the Masked BPS was used and global variable x_0 was masked with probability 0.5 at each refresh event. Up to 12 vCPUs were used, when the number of groups, G exceeded 15 then multiple independent sub-graphs were assigned to the same process, hence the decreasing efficiency as G increased.

Table 3: Logistic Regression, ESS/s $\times 10^{-2}$

d	Method	$G = 25$	50	75	100
10	Local BPS	**35.27**	**21.61**	12.69	11.45
	Masked BPS	18.13	21.39	**26.96**	**21.03**
25	Local BPS	9.68	6.57	4.69	4.50
	Masked BPS	**18.58**	**28.44**	**20.60**	**17.76**
50	Local BPS	3.25	2.58	2.52	2.23
	Masked BPS	**19.75**	**16.61**	**16.41**	**13.95**

B.3 Dynamic Bradley-Terry

The Dynamic Bradley-Terry model again consists of Gaussian and logistic factors, the event time proposals were sampled using the time-scale transformation and alias table thinning procedures similar to the previous experiments. Dynamic factorization was achieved in the same way as the state-space model example.