

# Recycling Intermediate Steps to Improve Hamiltonian Monte Carlo

Akihiko Nishimura\* and David Dunson\*

**Abstract.** Hamiltonian Monte Carlo (HMC) and related algorithms have become routinely used in Bayesian computation. In this article, we present a simple and provably accurate method to improve the efficiency of HMC and related algorithms with essentially no extra computational cost. This is achieved by *recycling* the intermediate states along simulated trajectories of Hamiltonian dynamics. Standard algorithms use only the end points of trajectories, wastefully discarding all the intermediate states. Compared to the alternative methods for utilizing the intermediate states, our algorithm is simpler to apply in practice and requires little programming effort beyond the usual implementations of HMC and related algorithms. Our algorithm applies straightforwardly to the no-U-turn sampler, arguably the most popular variant of HMC. Through a variety of experiments, we demonstrate that our recycling algorithm yields substantial computational efficiency gains.

**Keywords:** Bayesian inference, Hamiltonian Monte Carlo, Markov chain Monte Carlo, multi-proposal, Rao-Blackwellization.

## 1 Introduction

Markov chain Monte Carlo is routinely used to generate samples from posterior distributions. While specialized algorithms exist for restricted model classes, general-purpose algorithms are often inefficient and scale poorly in the number of parameters. Originally proposed by Duane et al. (1987) and popularized in the statistics community through the works of Neal (1996, 2010), Hamiltonian Monte Carlo promises a better scalability (Neal, 2010; Beskos et al., 2013) and has enjoyed wide-ranging successes as one of the most reliable approaches in general settings (Gelman et al., 2013; Kruschke, 2014; Monahan et al., 2016). Stan and PyMC software packages take advantage of this generality and performance (Stan Development Team, 2015; Salvatier et al., 2016).

Given a parameter  $\boldsymbol{\theta} \sim \pi_{\boldsymbol{\theta}}(\cdot)$  of interest, HMC introduces an auxiliary *momentum* variable  $\mathbf{p}$  and defines a distribution  $\pi(\cdot) = \pi_{\boldsymbol{\theta}}(\cdot) \times \mathcal{N}(\mathbf{0}, \mathbf{M})$  on the augmented parameter space  $(\boldsymbol{\theta}, \mathbf{p})$  with a *mass matrix*  $\mathbf{M}$ . A proposal is generated by simulating trajectories of *Hamiltonian dynamics* where the evolution of the state  $(\boldsymbol{\theta}, \mathbf{p})$  is governed by a differential equation:

$$\frac{d\boldsymbol{\theta}}{dt} = \mathbf{M}^{-1}\mathbf{p}, \quad \frac{d\mathbf{p}}{dt} = \nabla \log \pi_{\boldsymbol{\theta}}(\boldsymbol{\theta}). \quad (1.1)$$

---

\*Department of Biomathematics, University of California Los Angeles, Los Angeles, CA 90095, USA, [akihiko4@ucla.edu](mailto:akihiko4@ucla.edu)

†Department of Statistical Science, Duke University, Durham, NC 27708, USA, [dunson@duke.edu](mailto:dunson@duke.edu)

Proposals generated by this mechanism can be far away from the current state and yet accepted with high probability. In case  $\mathbf{M} = \mathbf{I}$  and  $\boldsymbol{\theta} \in \mathbb{R}^2$ , the solution trajectory of (1.1) coincides with the motion of a frictionless puck sliding over a surface of height  $-\log \pi(\boldsymbol{\theta})$  (Neal, 2010). The puck feels “push” in the direction of the gradient  $\nabla \log \pi(\boldsymbol{\theta})$ , pointing toward higher values of  $\log \pi(\boldsymbol{\theta})$ . The momentum evolves accordingly but also gives the puck tendency to keep moving in the same direction, helping HMC proposals explore the parameter space in an informed manner.

In practice, the analytical solution to (1.1) is rarely available and a trajectory  $(\boldsymbol{\theta}(t), \mathbf{p}(t))$  for  $0 \leq t \leq \tau$  is approximated by taking  $K \approx \tau/\epsilon$  steps of a *leap-frog* scheme with stepsize  $\epsilon$ , where each step  $F_\epsilon : (\boldsymbol{\theta}_0, \mathbf{p}_0) \rightarrow (\boldsymbol{\theta}_1, \mathbf{p}_1)$  is defined via the relations

$$\begin{aligned} \mathbf{p}_{1/2} - \mathbf{p}_0 &= \frac{\epsilon}{2} \nabla \log \pi_{\boldsymbol{\theta}}(\boldsymbol{\theta}_0) \\ \boldsymbol{\theta}_1 - \boldsymbol{\theta}_0 &= \epsilon \mathbf{M}^{-1} \mathbf{p}_{1/2} \\ \mathbf{p}_1 - \mathbf{p}_{1/2} &= \frac{\epsilon}{2} \nabla \log \pi_{\boldsymbol{\theta}}(\boldsymbol{\theta}_1). \end{aligned} \tag{1.2}$$

The approximate solution  $F_\epsilon^K(\boldsymbol{\theta}_0, \mathbf{p}_0) \approx (\boldsymbol{\theta}(\tau), \mathbf{p}(\tau))$  no longer has the distribution  $\pi(\cdot)$ , but can be used as a Metropolis proposal (Metropolis et al., 1953). Efficiency of HMC depends critically on a choice of  $\epsilon$ ,  $\tau$ , and  $\mathbf{M}$ , but well-established approaches exist for tuning these parameters (Andrieu and Thoms, 2008; Neal, 2010; Wang et al., 2013; Hoffman and Gelman, 2014; Stan Development Team, 2015).

Current practice uses the last state  $F_\epsilon^K(\boldsymbol{\theta}_0, \mathbf{p}_0)$  as a proposal and discards all the intermediate values  $F_\epsilon^k(\boldsymbol{\theta}_0, \mathbf{p}_0)$  for  $k < K$ . As we will show, this is wasteful since the intermediate values can be *recycled* to generate additional samples from posterior distributions. Figure 1, to be explained in detail later, illustrates the benefit of recycling. The recycling algorithm only requires quantities that have already been sampled or computed, so there is essentially no extra computational cost. Our proposed recycling approach can also be applied directly to a wide variety of modified HMC algorithms (Neal, 2010; Girolami and Calderhead, 2011; Pakman and Paninski, 2013, 2014; Lan et al., 2014; Shahbaba et al., 2014; Fang et al., 2014; Zhang et al., 2016; Lu et al., 2016). Extensions to more complex variants are also possible, including the No-U-Turn-Sampler (NUTS) (Hoffman and Gelman, 2014; Stan Development Team, 2015).

Our algorithm is distinguished by its simplicity and generality compared to alternative algorithms for utilizing the intermediate values of HMC (Neal, 1994; Calderhead, 2014; Bernton et al., 2015). Under our framework, one can typically implement an HMC variant as usual and simply add several lines of code to recycle the intermediate values using the familiar acceptance and rejection probabilities. On the other hand, the algorithm of Calderhead (2014) and its Rao-Blackwellization by Bernton et al. (2015) require a trajectory to be simulated forward and backward in a symmetric manner to satisfy the *super-detailed balance* condition (Frenkel, 2004; Tjelmeland, 2004). The proposal must then be followed by the acceptance-rejection step using the generalized Metropolis-Hastings algorithm (Calderhead, 2014) or assignment of appropriate weights to the intermediate values (Bernton et al., 2015). Importantly, these algorithms do not apply to NUTS, arguably the most popular variant of HMC. This is because NUTS

yields a variable number of intermediate states and does not constitute a multi-proposal scheme necessary for using their algorithms.

The underlying idea behind our algorithm is most similar to Neal (1994) who realized that, in the variant of HMC that uses a collection of states in computing the acceptance probability, those states can be re-used when computing the posterior summaries through conditional expectation. Our theory is more general, however, and translates into methods to improve a variety of multi-proposal algorithms (Section 3).

## 2 Recycled Hamiltonian Monte Carlo

The following non-standard HMC algorithm accepts or rejects each of the intermediate values, enabling recycling of these samples.

**Algorithm 1** (Recycled HMC). *Generate random variables  $\{(\boldsymbol{\theta}_k^{(i)}, \mathbf{p}_k^{(i)})\}$ ,  $k = 0, 1, \dots, K\}_{i \geq 1}$  so that the sequence  $\{(\boldsymbol{\theta}_0^{(i)}, \mathbf{p}_0^{(i)})\}_{i \geq 1}$  forms a Markov chain with transition rule  $(\boldsymbol{\theta}_0^{(i)}, \mathbf{p}_0^{(i)}) \rightarrow (\boldsymbol{\theta}_0^{(i+1)}, \mathbf{p}_0^{(i+1)})$  as follows:*

1. Set  $\boldsymbol{\theta}_0 = \boldsymbol{\theta}_0^{(i)}$  and draw a random momentum  $\mathbf{p}_0 \sim \mathcal{N}(\mathbf{0}, \mathbf{M})$ .
2. Simulate a trajectory via the leapfrog steps as in (1.2) and set  $(\boldsymbol{\theta}_k^*, \mathbf{p}_k^*) = F_\epsilon^k(\boldsymbol{\theta}_0, \mathbf{p}_0)$  for  $k = 1, \dots, K$ .
3. Accept or reject each state along the trajectory; for  $k = 1, \dots, K$ , set  $(\boldsymbol{\theta}_k^{(i+1)}, \mathbf{p}_k^{(i+1)}) = (\boldsymbol{\theta}_k^*, \mathbf{p}_k^*)$  with probability

$$\min \left\{ 1, \frac{\pi(\boldsymbol{\theta}_k^*, \mathbf{p}_k^*)}{\pi(\boldsymbol{\theta}_0, \mathbf{p}_0)} \right\} \tag{2.1}$$

and  $(\boldsymbol{\theta}_k^{(i+1)}, \mathbf{p}_k^{(i+1)}) = (\boldsymbol{\theta}_0, \mathbf{p}_0)$  otherwise.

4. Update the starting point of the next HMC iteration:  $(\boldsymbol{\theta}_0^{(i+1)}, \mathbf{p}_0^{(i+1)}) = (\boldsymbol{\theta}_K^{(i)}, \mathbf{p}_K^{(i)})$ .

The transition rule  $(\boldsymbol{\theta}_0^{(i)}, \mathbf{p}_0^{(i)}) \rightarrow (\boldsymbol{\theta}_0^{(i+1)}, \mathbf{p}_0^{(i+1)})$  above coincides with that of the standard HMC algorithm. The empirical measure  $N^{-1} \sum_{i=1}^N \delta_{(\boldsymbol{\theta}_0^{(i)}, \mathbf{p}_0^{(i)})}(\cdot)$  thus converges weakly to the target distribution. The following theorem, which is a consequence of a more general theory given in the next section, shows that the intermediate samples can actually be recycled as valid draws from the target distribution. The index  $k$  runs from 1 to  $K$  as  $(\boldsymbol{\theta}_0^{(i+1)}, \mathbf{p}_0^{(i+1)}) = (\boldsymbol{\theta}_K^{(i)}, \mathbf{p}_K^{(i)})$  and  $k = 0$  is redundant.

**Theorem 1.** *If  $(\boldsymbol{\theta}_k^{(i)}, \mathbf{p}_k^{(i)})$  for  $k = 1, \dots, K$  are generated as in Algorithm 1, then*

$$\frac{1}{NK} \sum_{i=1}^N \sum_{k=1}^K \delta_{(\boldsymbol{\theta}_k^{(i)}, \mathbf{p}_k^{(i)})}(\cdot) \xrightarrow{w} \pi(\cdot) \text{ as } N \rightarrow \infty, \tag{2.2}$$

where  $\xrightarrow{w}$  denotes the weak convergence of a measure.

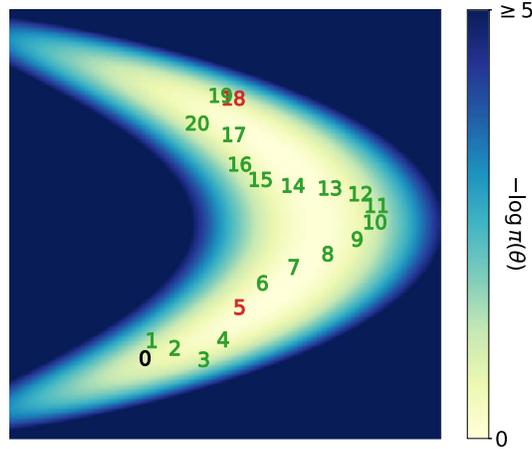


Figure 1: Visual demonstration of the recycling algorithm applied during one iteration of HMC on a 2-D banana-shaped target distribution. The numbers  $k = 0, 1, \dots, 20$  indicate the location  $(\theta_k, \mathbf{p}_k)$  of a simulated trajectory after  $k$  leapfrog steps. Red color at  $k = 5$  and  $k = 18$  indicates rejection of the intermediate states. Green color indicates the successfully recycled (i.e. accepted) intermediate states. Most of the intermediate states have high acceptance rates as expected from the property of Hamiltonian dynamics.

Fig. 1 illustrates the recycling procedure. The Metropolis type acceptance-rejection of (2.1) is applied to each intermediate state  $F_\epsilon^k(\theta_0^{(i)}, \mathbf{p}_0^{(i)})$ . Calculating these acceptance probabilities typically takes little additional computational time; the unnormalized target densities at the intermediate values are either already computed in common variants of HMC (Neal, 2010; Hoffman and Gelman, 2014) or can be obtained cheaply as a by-product of computing the gradients  $\nabla \log \pi_\theta$ .

Algorithm 1 uses the fixed path length  $K$ , but it is often desirable to randomize the number of steps to avoid periodic behavior in the trajectories of (1.1) (Neal, 2010). Recycling is justified under this setting as well:

**Theorem 2.** *Consider a modified version of Algorithm 1 in which at the  $i$ -iteration 1) the trajectory is simulated for a random path length  $L^{(i)} \sim \pi_L(\cdot)$  with  $L^{(i)} \leq K$  and 2) the starting point for the next HMC iteration is set as  $(\theta_0^{(i+1)}, \mathbf{p}_0^{(i+1)}) = (\theta_{L^{(i)}}^{(i)}, \mathbf{p}_{L^{(i)}}^{(i)})$ . Then we have*

$$\frac{1}{\sum_{i=1}^N L^{(i)}} \sum_{i=1}^N \sum_{k=1}^{L^{(i)}} \delta_{(\theta_k^{(i)}, \mathbf{p}_k^{(i)})}(\cdot) \xrightarrow{w} \pi(\cdot) \text{ as } N \rightarrow \infty. \tag{2.3}$$

### 3 Theory behind recycling algorithm

The validity of recycled HMC as in Theorem 1 and 2 follows from a more general principle below.

**Theorem 3.** Let  $P_k(\cdot|\cdot)$  for  $k = 0, \dots, K$  be transition kernels with a common stationary measure  $\pi(\cdot)$  and suppose  $P_0(\cdot|\cdot)$  is uniquely ergodic.<sup>1</sup> Consider a Markov chain  $\{\mathbf{z}^{(i)}\}_{i \geq 1}$  on a product space  $\mathbf{z} = (z_0, \dots, z_K)$  whose transition probability  $\mathbf{z} \rightarrow \mathbf{z}^*$  only depends on the coordinate  $z_0$  i.e.

$$P(\mathbf{z}_0^*, \dots, \mathbf{z}_K^* | z_0, \dots, z_K) = P(\mathbf{z}_0^*, \dots, \mathbf{z}_K^* | z_0) \tag{3.1}$$

and has the marginal densities

$$\int P(\mathbf{z}_0^*, \dots, \mathbf{z}_K^* | z_0) dz_{-k}^* = P_k(\mathbf{z}_k^* | z_0) \tag{3.2}$$

where  $\mathbf{z}_{-k}^* = (z_0^*, \dots, z_{k-1}^*, z_{k+1}^*, \dots, z_K^*)$  for  $k = 0, \dots, K$ . Then the following result holds:

$$\frac{1}{NK} \sum_{i=1}^N \sum_{k=1}^K \delta_{\mathbf{z}_k^{(i)}}(\cdot) \xrightarrow{w} \pi(\cdot) \text{ as } N \rightarrow \infty. \tag{3.3}$$

Additionally, the Markov chain  $\{\mathbf{z}^{(i)}\}_{i \geq 1}$  is geometrically (or uniformly) ergodic if  $P_0(\cdot|\cdot)$  is geometrically (or uniformly) ergodic.

*Proof.* It is easy to verify that the Markov chain  $\mathbf{z}^{(1)}, \mathbf{z}^{(2)}, \dots$  has a stationary distribution

$$\pi^*(\cdot) = \int P(\cdot | z_0) \pi(z_0) dz_0. \tag{3.4}$$

By the assumption (3.2), the marginal  $\pi^*(z_k)$  coincides with  $\pi(z_k)$  for all  $k = 0, \dots, K$ . Once we establish the unique ergodicity of the chain  $\{\mathbf{z}^{(i)}\}_{i \geq 1}$ , therefore, the conclusion (3.3) follows by averaging the coordinates  $\mathbf{z}_1, \dots, \mathbf{z}_k$  of the empirical measure  $N^{-1} \sum_{i=1}^N \delta_{\mathbf{z}^{(i)}}$ .

To see that  $\pi^*(\cdot)$  is the unique stationary measure, suppose  $\tilde{\pi}^*(\cdot)$  is another stationary measure of  $P(\cdot|\cdot)$ . This means that, by the assumption (3.1),

$$\tilde{\pi}^*(\cdot) = \int P(\cdot | z_0) \tilde{\pi}^*(z_0) dz_0. \tag{3.5}$$

In particular, the marginal  $\tilde{\pi}^*(z_0^*)$  satisfies  $\tilde{\pi}^*(z_0^*) = \int P_0(z_0^* | z_0) \tilde{\pi}^*(z_0) dz_0$  by the assumption (3.2). The unique ergodicity of  $P_0(\cdot|\cdot)$  then implies  $\tilde{\pi}^*(z_0) = \pi(z_0)$ . Substituting this equality into (3.5) establishes  $\tilde{\pi}^*(\cdot) = \pi^*(\cdot)$  and hence the unique ergodicity of the chain  $\{\mathbf{z}^{(i)}\}_{i \geq 1}$ .

We turn to the proof of a convergence rate (geometric or uniform ergodicity) of the chain  $\{\mathbf{z}^{(i)}\}_{i \geq 1}$  under the corresponding assumption on  $P_0(\cdot|\cdot)$ . For the conditional distribution of  $\mathbf{z}^{(n)} | \mathbf{z}_0^{(0)}$ , we have

$$\left| \mathbb{P}(\mathbf{z}^{(n)} \in A | \mathbf{z}_0^{(0)}) - \pi^*(A) \right| = \left| \int P(A | z'_0) \left( P_0^n(\mathbf{z}'_0 | \mathbf{z}_0^{(0)}) - \pi(\mathbf{z}'_0) \right) dz'_0 \right|. \tag{3.6}$$

---

<sup>1</sup>A transition kernel (or a Markov chain) with a unique stationary measure is called *uniquely ergodic*. The uniqueness of a stationary measure implies ergodicity by the ergodic decomposition theorem (Kallenberg, 2002).

It follows that

$$\left\| \mathbb{P}(\mathbf{z}^{(n)} \in \cdot \mid \mathbf{z}_0^{(0)}) - \pi^*(\cdot) \right\|_{\text{tv}} \leq \left\| P_0^n(\cdot \mid \mathbf{z}_0^{(0)}) - \pi(\cdot) \right\|_{\text{tv}}$$

where  $\|\cdot\|_{\text{tv}}$  denotes a total variation norm. Hence the chain  $\{\mathbf{z}^{(i)}\}_{i \geq 1}$  inherits the convergence rate of  $P_0(\cdot \mid \cdot)$ .  $\square$

Theorem 3 has a subtle but important difference from “composition sampling,” in which one would first generate a Markov chain  $\{\mathbf{z}_0^{(i)}\}_{i \geq 0}$  and then sample  $(\mathbf{z}_1^{(i+1)}, \dots, \mathbf{z}_K^{(i+1)})$  from a conditional distribution  $\pi^*(\cdot \mid \mathbf{z}_0^{(i)})$ . For a Markov chain generated as in Theorem 3, the conditional distribution  $\mathbf{z}_1^{(i+1)}, \dots, \mathbf{z}_K^{(i+1)} \mid \mathbf{z}_0^{(i)}$  may have dependency on  $\mathbf{z}_0^{(i+1)}$ . This additional flexibility is critical for the recycling algorithms presented in this article. In particular, Theorem 3 reduces to Theorem 1 when the transition kernel  $P_k(\cdot \mid \cdot)$  is constructed as one iteration of HMC with  $k$  leapfrog steps for  $k \geq 1$  and  $P_0(\cdot \mid \cdot)$  as that with  $K$  steps.

Theorem 2 is justified by an extension of Theorem 3, which shows that a recycling algorithm applies even when the number of states generated by a multi-proposal scheme varies from one iteration to another. Since in Theorem 2 the number of recyclable samples varies according to the random path length  $L^{(i)}$ 's, the general theory behind Theorem 2 relies on the framework of a Markov chain in a trans-dimensional parameter space (Hastie and Green, 2012). The precise statement is given as Theorem 4 along with a proof in Appendix A (Nishimura and Dunson, 2019).

The general formulation of the recycling algorithm as in Theorem 3 and 4 is of practical value for any Markov chain Monte Carlo (MCMC) algorithm that simultaneously yields multiple valid transition kernels  $P_k(\cdot \mid \cdot)$ 's. Indeed, in many variants of HMC (Neal, 2010; Girolami and Calderhead, 2011; Shahbaba et al., 2014; Fang et al., 2014), a proposal is generated by computing a long trajectory whose intermediate steps constitute valid proposal states that can be all recycled by simply adding acceptance-rejection steps as in Algorithm 1. Our theory also provides an alternate and simpler justification of the algorithms by Calderhead (2014) and Bernton et al. (2015) as shown in Appendix B. Our recycling algorithm can also be applied to more complex proposal generation mechanisms as we illustrate in the next section.

## 4 Recycled no-U-turn-sampler

No-U-turn sampler (NUTS) of Hoffman and Gelman (2014) automates choice of path lengths by simulating each trajectory of Hamiltonian dynamics until it starts moving back towards the starting point, a criteria they termed the *U-turn* condition. The length of a trajectory is recursively doubled forward or backward in a randomly chosen direction. This generates a trajectory of length  $2^d$  endowed with a binary tree structure, where  $d$  denotes the depth at which the U-turn condition occurs.

Unlike the simpler trajectory simulation procedure behind HMC, the trajectory doubling procedure of NUTS does not yield a sequence of valid intermediate proposals. In

particular, the empirical distribution does not converge to the correct target distribution if we naively recycle all the intermediate states of NUTS as in Algorithm 1. A simple recycling algorithm for NUTS can nonetheless be devised by taking advantage of the fact below. Instead of the Metropolis acceptance-rejection procedure, NUTS determines acceptable states along a simulated trajectory using a slice sampling approach via an auxiliary slicing variable  $u$ .

**Fact 1.** *The following transition rule  $(\boldsymbol{\theta}_0, \mathbf{p}_0) \rightarrow (\boldsymbol{\theta}^*, \mathbf{p}^*)$  preserves the target distribution  $\pi(\cdot)$ . Let  $\mathcal{T} = \mathcal{T}(\boldsymbol{\theta}_0, \mathbf{p}_0)$  denote a collection of  $2^d$  states generated by one NUTS iteration from the initial state  $(\boldsymbol{\theta}_0, \mathbf{p}_0)$ , including  $(\boldsymbol{\theta}_0, \mathbf{p}_0)$  itself. Generate  $u \sim \text{Unif}([0, \pi(\boldsymbol{\theta}_0, \mathbf{p}_0)])$  and sample  $(\boldsymbol{\theta}^*, \mathbf{p}^*)$  uniformly from the collection of acceptable states*

$$\mathcal{A} = \mathcal{A}(\mathcal{T}, u) = \{(\boldsymbol{\theta}, \mathbf{p}) \in \mathcal{T} \mid \pi(\boldsymbol{\theta}, \mathbf{p}) > u\}. \tag{4.1}$$

The algorithmic details behind construction of the set  $\mathcal{A}$  are complex. Since it is not essential for understanding our recycling approach, we refer the readers to Hoffman and Gelman (2014). The discussions there also justify the above transition rule.

Fact 1 motivates the following algorithm for utilizing the intermediate states generated during each iteration of NUTS.

**Algorithm 2** (Simple Recycled NUTS). *Run NUTS to generate a sequence of random variables  $\{(\boldsymbol{\theta}_0^{(i)}, \mathbf{p}_0^{(i)})\}_{i \geq 1}$ . Additionally at each iteration of NUTS, generate  $\{(\boldsymbol{\theta}_k^{(i)}, \mathbf{p}_k^{(i)}), k = 1, \dots, K\}$  by sampling  $K$  variables without replacement from the acceptable states  $\mathcal{A}(\mathcal{T}(\boldsymbol{\theta}_0^{(i-1)}, \mathbf{p}_0^{(i-1)}))$  as in (4.1).*

Fact 1 tells us that the transition  $(\boldsymbol{\theta}_0^{(i)}, \mathbf{p}_0^{(i)}) \rightarrow (\boldsymbol{\theta}_k^{(i)}, \mathbf{p}_k^{(i)})$  preserves the target distribution  $\pi(\cdot)$  for each  $k = 1, \dots, K$ . Algorithm 2 is thus justified with a straightforward application of Theorem 3.

Algorithm 2 captures the main idea behind our recycling algorithm; however, it is actually more statistically efficient to sample  $(\boldsymbol{\theta}_1^{(i)}, \mathbf{p}_1^{(i)}), \dots, (\boldsymbol{\theta}_K^{(i)}, \mathbf{p}_K^{(i)})$  from  $\mathcal{A}(\mathcal{T}(\boldsymbol{\theta}_0^{(i-1)}, \mathbf{p}_0^{(i-1)}))$  so that they are evenly spread along a NUTS trajectory. Such a sampling scheme can be implemented in a simple and memory efficient manner — without storing all the intermediate states in memory — by taking advantage of the binary tree structure of a NUTS trajectory. This is described in Appendix C.

When we are not constrained by memory, the following Rao-Blackwellized version of recycled NUTS allows us to simply collect and use all the acceptable states of each NUTS iteration by assigning appropriate weights.

**Algorithm 3** (Rao-Blackwellized Recycled NUTS). *Denote the collection of acceptable states from the  $i$ -th iteration of NUTS by  $\mathcal{A}_i = \{(\boldsymbol{\theta}_k^{(i)}, \mathbf{p}_k^{(i)}), k = 1, \dots, |\mathcal{A}_i|\}$ . Return the samples  $\{(\boldsymbol{\theta}_k^{(i)}, \mathbf{p}_k^{(i)}), k = 1, \dots, |\mathcal{A}_i|\}$  with weight  $\propto |\mathcal{A}_i|^{-1}$  for  $i = 1, \dots, N$  as the draws from the target distribution, yielding an empirical measure:*

$$\frac{1}{N} \sum_{i=1}^N \frac{1}{|\mathcal{A}_i|} \sum_{k=1}^{|\mathcal{A}_i|} \delta_{(\boldsymbol{\theta}_k^{(i)}, \mathbf{p}_k^{(i)})}(\cdot).$$

The validity of Algorithm 3 follows simply by taking an expectation over the sampling step  $(\boldsymbol{\theta}_k^{(i)}, \mathbf{p}_k^{(i)}) \sim \text{Uniform}(\mathcal{A}_i)$  of Algorithm 2.

## 5 Numerical results

We demonstrate the benefit of recycling using four test cases: three taken from Hoffman and Gelman (2014) and one from Girolami and Calderhead (2011). We focus on these well-established test cases as guaranteeing a rapid convergence and mixing of HMC in full generality is an active area of research (Livingstone et al., 2016, 2017; Mangoubi and Smith, 2017). In all our simulations we chose the stepsizes  $\epsilon$  such that the corresponding average acceptance rates are approximately 70%, as values between 60% and 80% are typically considered optimal (Neal, 2010; Beskos et al., 2013). The dual averaging algorithm of Hoffman and Gelman (2014) was used to find such stepsizes. The choice of path lengths  $\tau^{(i)} = \epsilon L^{(i)}$  for HMC is discussed within the individual test cases below. The identity mass matrix  $\mathbf{M} = \mathbf{I}$  was used in all our simulations except when investigating the use of recycling in mass matrix tuning (see Section 5.1).

### 5.1 Metrics for computational and memory efficiency of recycling

#### Effective sample sizes for mean, variance, and quantile estimators

In comparing the algorithms with and without recycling, we use effective sample sizes (ESS) as a commonly used measure of computational efficiency of Monte Carlo algorithms (Brooks et al., 2011). ESS for a statistics of interest  $\mathbb{E}[f(\boldsymbol{\theta})]$  is defined via the relation

$$\text{Var}\left(N^{-1} \sum_{i=1}^N f(\boldsymbol{\theta}^{(i)})\right) = \text{ESS}^{-1} \text{Var}\left(f(\boldsymbol{\theta}^{(1)})\right), \quad (5.1)$$

comparing MCMC samples to the equivalent number of i.i.d. samples. The standard definition above applies only to estimators of the form  $N^{-1} \sum_{i=1}^N f(\boldsymbol{\theta}^{(i)})$  for a real-valued function  $f$ , so we extend it to a more complex estimator  $F : \{\boldsymbol{\theta}^{(i)}\}_{i=1}^N \rightarrow \mathbb{R}$  of a quantity  $\mathbb{E}[f(\boldsymbol{\theta})]$  by defining

$$\text{ESS}_F\left(\{\boldsymbol{\theta}^{(i)}\}_{i=1}^N\right) = N \frac{\text{MSE}\left(F\left(\left\{\boldsymbol{\theta}^{*(i)} \stackrel{\text{i.i.d.}}{\sim} \pi(\cdot)\right\}\right)\right)}{\text{MSE}\left(F\left(\{\boldsymbol{\theta}^{(i)}\}\right)\right)} \quad (5.2)$$

where  $\text{MSE}(\cdot)$  denotes the mean squared error of an estimator. The definition (5.2) agrees with the standard one when  $F(\{\boldsymbol{\theta}^{(i)}\}_{i=1}^N) = N^{-1} \sum_{i=1}^N f(\boldsymbol{\theta}^{(i)})$ .

In each of our examples, we compute the ratios of ESS with and without recycling for the mean, variance, and 97.5% quantile estimator along each model parameter. The ESS ratio is obtained from the ratio of MSE using the relation (5.2). The MSE of the Monte Carlo estimates are estimated by running 400 chains independently for 3,200 iterations starting from stationarity.<sup>2</sup> The number of iterations here is chosen to roughly

<sup>2</sup>To obtain stationary samples, we use the last sample of the long ground-truth NUTS chain as the starting point and run an additional 100 iterations to ensure that the starting points of the 400 chains are independent. We assessed sampling efficiency of NUTS via ESS to ensure that 100 iterations are more than enough to yield an independent sample.

agree with the typical use cases of HMC — for example, Stan’s default setting generates 1,000 NUTS samples after 1,000 warm-up iterations (Stan Development Team, 2015). We find in many cases, however, that the magnitudes of ESS gains from recycling are independent of the chain lengths (Appendix D). In computing the MSE of a Monte Carlo estimate, we need to have a precise value for the estimated quantity. Since the analytical expressions are unavailable in our test cases (except for the first synthetic one), we run a long NUTS chain of  $10^7$  iterations (after  $10^3$  burn-ins) to obtain the “ground truths” that are orders of magnitudes more accurate than the individual Monte Carlo estimates from the chains of length 3,200.

### Benefits of recycling in estimating covariance structures

We also assess whether recycling helps estimate the covariance structure of the target distribution. To this end, we compute the top eigenvalue and eigenvector of the empirical covariance matrix for each chain. We then calculate the angle between the empirical eigenvector and the plane spanned by the  $\ell$  true leading principal components. This angle should be close to 0 when the eigenvector is estimated well. To ensure identifiability of the direction, we choose  $\ell = \min\{j : \sigma_j^2 < \sigma_1^2/2\}$  in all our simulations where  $\sigma_j^2$  denotes the  $j$ th largest eigenvalue of the true covariance matrix.

As an alternate and more holistic evaluation of covariance estimation with a practical application, we further investigate the utility of recycling during the tuning phase of HMC/NUTS. Often, initial iterations of HMC/NUTS are used to estimate the covariance matrix of the target  $\Sigma = \text{Var}(\theta)$ , which can then be used to accelerate HMC/NUTS by setting the mass matrix  $M = \hat{\Sigma}^{-1}$  (Neal, 2010; Stan Development Team, 2015). If recycling improves the covariance estimator  $\hat{\Sigma}$  during the tuning phase, later HMC/NUTS iterations will be faster and mix better.

To quantify the benefit of recycling in this setting, we estimate the covariance with and without recycling during the tuning phase, and then run two independent chains with the two covariance estimators to compare their ESSs. Recycling is only applied during the tuning phase for covariance estimation. In carrying out this experiment, we follow the default settings of Stan for tuning the stepsize and mass matrix. First, 50 iterations of the dual-averaging algorithm are run to tune the stepsize with the identity mass matrix, followed by  $N_{\text{adap}}$  iterations with a fixed stepsize to estimate the covariance matrix, and finally another 75 iterations of dual-averaging to re-adjust the stepsize with the tuned mass matrix. After the covariance estimation phase with  $N_{\text{adap}}$  iterations, we set  $M^{-1} = \hat{\Sigma}$  where

$$\hat{\Sigma} = \frac{N_{\text{adap}}}{5 + N_{\text{adap}}} \hat{\Sigma}_{\text{emp}} + \frac{5}{5 + N_{\text{adap}}} 10^{-3} \cdot \mathbf{I}, \quad (5.3)$$

with  $\hat{\Sigma}_{\text{emp}}$  the empirical covariance matrix and  $\mathbf{I}$  the identity matrix. After the tuning phase, we run NUTS until the total number of gradient evaluations reaches  $10^4$ . This procedure is repeated 400 times and the ESS for each statistic is averaged across the repetitions. This experiment is not run for the models of Section 5.4 and 5.5 as the high-dimensionality of the parameter spaces make covariance estimation impractical.

**Memory and statistical efficiency trade-off**

We also study the relationship between the number of recycled samples and statistical efficiency. In particular, we demonstrate that it is not necessary to recycle all the intermediate steps to reap the benefit of recycling. This is relevant in a high dimensional parameter space where the amount of memory required to store the extra samples becomes substantial.<sup>3</sup> For long trajectories, there is substantial correlation among the intermediate states and we can expect that recycling a subset of the intermediate states may provide almost as much statistical efficiency as recycling all.

To quantify this, we first run the algorithm recycling all the intermediate states. We then repeatedly reduce  $K$ , the number of samples per iteration (recycled intermediate states plus the final state), by a factor of 2. In comparing the algorithms with and without recycling, we use the smallest  $K$  for which the ESS averaged across all the estimators is within 5% of that when recycling all the intermediate states. Section 5.6 investigates in more detail the relationship between the statistical efficiency and the number of recycled samples.

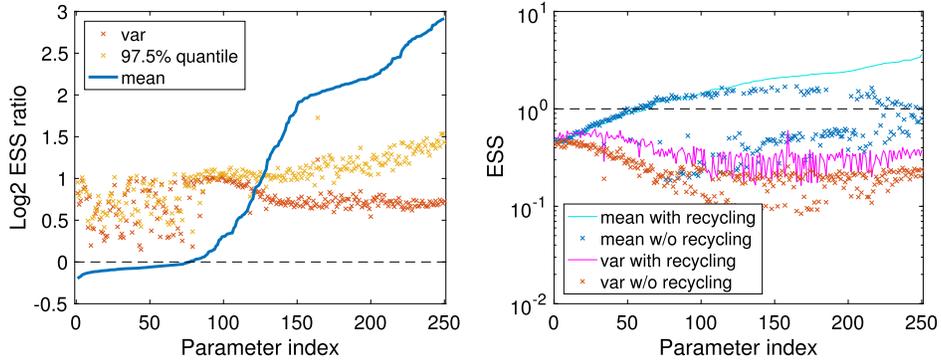
**5.2 Multivariate Gaussian**

The first test case is sampling from a 250-dimensional multivariate Gaussian  $\mathcal{N}(\mathbf{0}, \Sigma)$ , where  $\Sigma$  is drawn from a Wishart distribution with 250 degrees of freedom and mean equal to the identity matrix. A covariance matrix drawn from this distribution exhibits strong correlations, and in our case the ratio between the largest and smallest eigenvalue of  $\Sigma$  was approximately  $9.5 \times 10^4$ . Since HMC and NUTS with the identity mass matrix are invariant under rotations, we take  $\Sigma$  to be diagonal with  $\Sigma_{i,i} = \sigma_i^2$ , where  $\sigma_i^2$  corresponds to the  $i$ th smallest eigenvalue of the original covariance matrix. For the path length of HMC, we first found the smallest value of  $\tau$  for which the samples in the leading principal component direction are roughly independent. The typical practice would be then to jitter  $\tau^{(i)}$ 's within the range  $[0.9\tau, 1.1\tau]$  to avoid periodicity (Neal, 2010), but this still resulted in near perfect periodicity and hence poor mixing for some parameters. After some experiments, we found jittering  $\tau^{(i)}$  in the range  $[\tau/2, \tau]$  to provide decent mixing along all the coordinates.

Fig. 2a shows  $\log_2$  of the ratios between ESS of HMC with and without recycling. To facilitate the comparison of algorithms with and without recycling, the parameters are sorted in increasing order of the ESS ratios in mean estimation. Values above zero indicate superior performance of our recycling algorithm. The use of recycling improves, uniformly and substantially, estimation of the variances and quantiles: about 100% increase in ESS on average. Not all the mean estimators are improved by recycling, but the ones with worst ESS are significantly improved (Fig. 2b). Out of 251 recyclable samples generated on average from each iteration of HMC, we recycled  $251/8 \approx 31$  samples.

---

<sup>3</sup>In the stochastic volatility model of Section 5.4, for example, it requires 4 GB of memory to store 100 extra samples per iteration from a Markov chain of length 3,200 in a 3000-dimensional parameter space.



(a)  $\log_2$  ratios of ESS with recycling (numerator) and without recycling (denominator). The horizontal line at zero corresponds to no gain from recycling. (b) ESS per HMC iteration for the mean and variance estimators. The parameters with the smallest ESS are those with the worst mixings. The y-axis is in  $\log_{10}$  scale.

Figure 2: Performance comparison between HMC with and without recycling in estimating mean, variance, and quantiles for the Gaussian example.

The ratios of ESS in estimating the angle as well as the eigenvalues are shown in Fig. 3. We plot the ratios against the lengths of Markov chains. The direction of the principal component cannot be well estimated by shorter chains of lengths  $\sim 200$  even with recycling, but recycling conveys a substantial advantage as the chains are run longer. While here the ESS ratios vary substantially as a function of chain lengths, this dependence seems to be unique to this particular summary statistics — we show in Appendix D that, for the other statistics, the magnitudes of ESS gains from recycling are independent of the chain lengths.

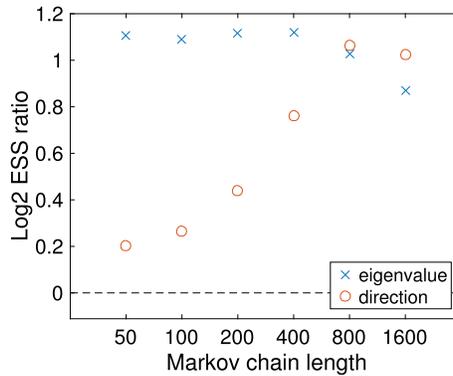


Figure 3: Performance comparison between HMC with and without recycling in estimating the direction and magnitude of the leading principal component for the covariance matrix in the Gaussian example.

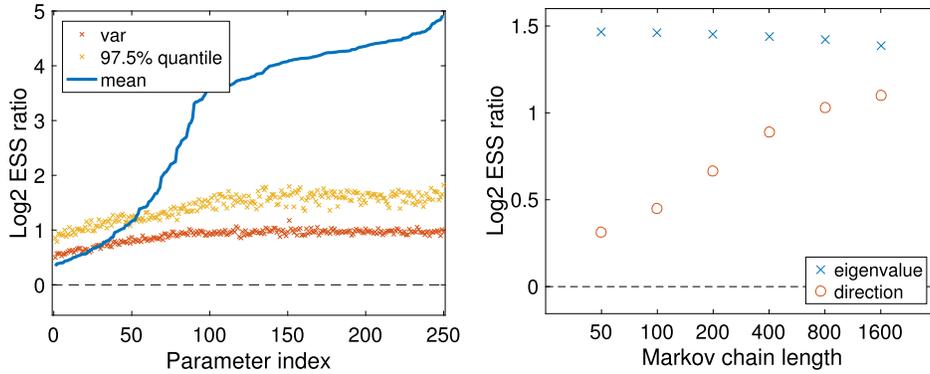


Figure 4: Performance comparison between NUTS with and without recycling for the Gaussian example.

Similar performance comparisons of NUTS with and without recycling are provided in Fig. 4. The average trajectory length was  $2^9 = 512$ , out of which  $2^4 - 1 = 15$  samples were recycled.

Lastly, the results of our covariance/mass matrix tuning experiment is summarized in Figure 5. Again, in this experiment recycling is only carried out during the tuning phase and the difference in ESS comes purely from difference in the accuracy of the covariance estimators. The benefit of recycling diminishes as  $N_{\text{adap}}$  increases as the covariance matrix can be adequately approximated without recycling and we found no advantage of recycling when  $N_{\text{adap}} \geq 800$ .

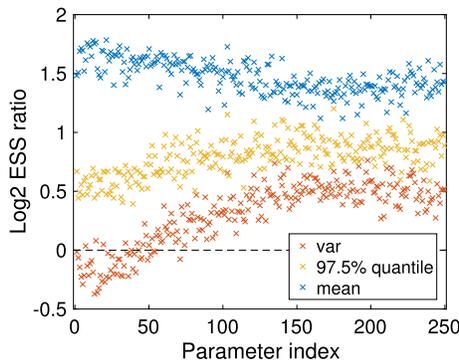


Figure 5:  $\log_2$  ratios of average ESS based on  $10^4$  gradient evaluations when the mass matrix is tuned with and without recycling for  $N_{\text{adap}} = 400$  in the Gaussian example.

### 5.3 Hierarchical Bayesian logistic regression

The second test case is a hierarchical Bayesian logistic regression model applied to the German credit data set available from the University of California Irvine Machine Learning Repository. Including two-way interaction terms and an intercept, there are 301 predictors and the regression coefficients  $\beta$  are given a  $\mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I})$  prior. A hyper-prior is placed on  $\sigma^2$ , which makes the posterior inference more challenging through the strong dependence between  $\sigma$  and  $\beta$ . We made one modification to the corresponding example in Hoffman and Gelman (2014) by defining our parameters to be  $(\log(\sigma), \beta)$  instead of  $(\sigma^2, \beta)$  since such a transformation of constrained variables has become standard (Stan Development Team, 2015). A default flat prior was placed on  $\sigma$ .

Performance comparisons as in the previous example are shown in Fig. 6, 7, and 8. For some parameters, recycling seems to produce little gains in terms of mean estimation

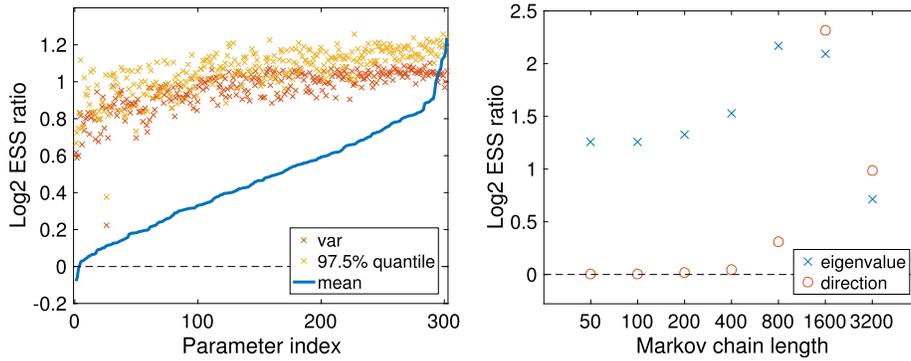


Figure 6: Performance comparison between HMC with and without recycling for the hierarchical logistic model.

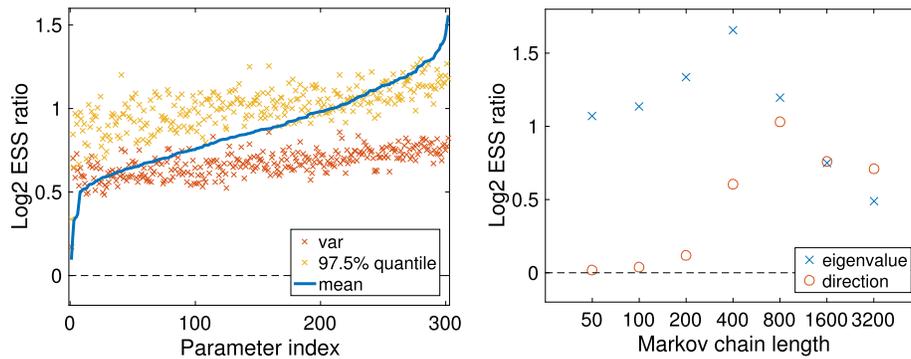


Figure 7: Performance comparison between NUTS with and without recycling for the hierarchical logistic model.

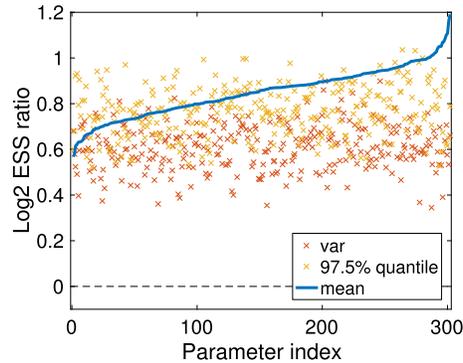


Figure 8: Comparison of average ESS based on  $10^4$  gradient evaluations between NUTS with a mass matrix tuned with and without recycling for  $N_{\text{adap}} = 500$  in the hierarchical logistic model.

but provides clear benefits in terms of variance and quantile estimation. In the mass matrix tuning experiment shown in Figure 8, we tried  $N_{\text{adap}} = 500, 1000, 2000$  and observed substantial improvement in the average ESS from recycling for  $N_{\text{adap}} \leq 1000$ .

For the path lengths for HMC, we first found the value  $\tau$  to maximize the normalized expected square jumping distance  $\tau^{-1/2} \mathbb{E} \|\boldsymbol{\theta}^{(i+1)}(\tau) - \boldsymbol{\theta}^{(i)}(\tau)\|$  as in Wang et al. (2013), then jittered each path length  $\tau^{(i)}$  in the range  $[0.9\tau, 1.1\tau]$ . The average trajectory length of HMC was 9 and all the intermediate states were recycled. The average trajectory length of NUTS was  $2^4 = 16$ , out of which 7 were recycled.

## 5.4 Stochastic volatility model

The third test case is a stochastic volatility (SV) model fit to a time series  $\mathbf{y}$  taken from the closing values of S&P 500 index for 3000 days ending on Dec 31st, 2015. The model is specified as follows:

$$\log \left( \frac{y_i}{y_{i-1}} \right) \sim \mathcal{N}(0, s_i^2), \quad 100 \log \left( \frac{s_i}{s_{i-1}} \right) \sim \mathcal{N}(0, \tau^{-1})$$

with priors  $s_0 \sim \text{Exp}(\text{mean} = 1/10)$  and  $\tau \sim \text{Gamma}(1/2, 1/2)$ . The observed value on Jan 2nd, 2008 was removed from the original data as this simple SV model could not fit this observation well. After integrating out  $\tau$  to accelerate mixing, we are left with a 3000 dimensional parameter space for  $\log s$ .

Performance comparisons as in the previous examples are shown in Fig. 9 and 10. The path length for HMC was chosen as in Section 5.3. On average, 44 samples out of 90 per iteration were recycled for HMC and 7 out of  $2^7 = 128$  were recycled for NUTS.

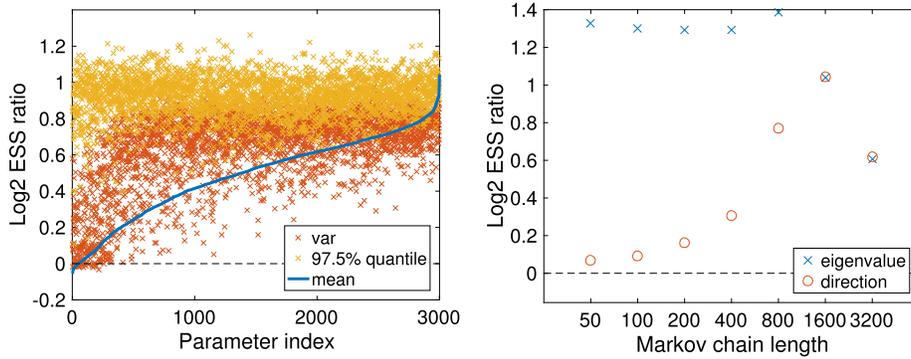


Figure 9: Performance comparison between HMC with and without recycling for the SV model.

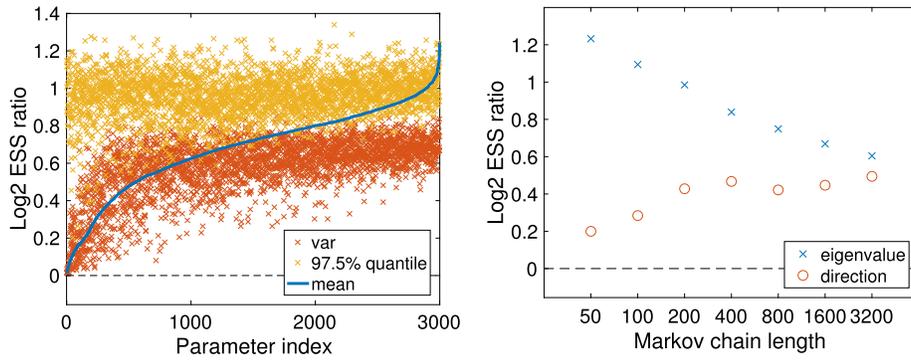


Figure 10: Performance comparison between NUTS with and without recycling for the SV model.

### 5.5 Log-Gaussian Cox point-process (Riemann manifold HMC)

The last test case is a log-Gaussian Cox point-process model from Girolami and Calderhead (2011), where they apply *Riemann manifold HMC* (RMHMC) to sample from the latent process  $\mathbf{x} \in \mathbb{R}^{4096}$  defined on a  $64 \times 64$  grid. The observation  $y_{ij}$  for  $i, j = 1, \dots, 64$  is assumed to follow

$$y_{ij} | x_{ij} \sim \text{Poisson}(\exp(x_{ij})/64^2). \tag{5.4}$$

The latent process  $\mathbf{x} | \mu, \sigma, \ell$  is given a Gaussian process prior with mean  $\mu \mathbf{1}$  and covariance

$$\text{cov}(x_{ij}, x_{i'j'}) = \sigma^2 \exp\left(\frac{-\sqrt{(i-i')^2 + (j-j')^2}}{64\ell}\right). \tag{5.5}$$

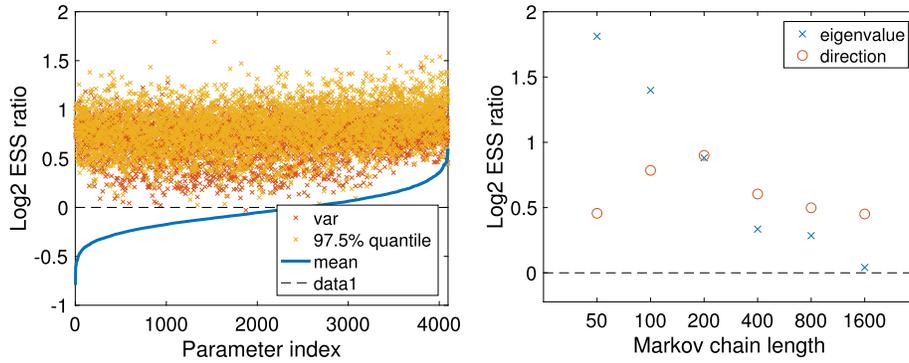


Figure 11: Performance comparison between RMHMC with and without recycling for the log-Gaussian Cox model.

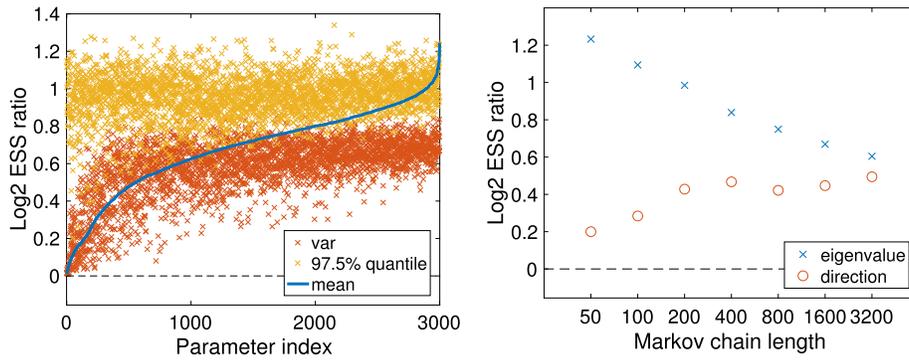


Figure 12: Performance comparison between NUTS with and without recycling for the log-Gaussian Cox model.

Following Girolami and Calderhead (2011), we fix the hyper-parameters at  $\ell = 1/33$ ,  $\sigma^2 = 1.91$ , and  $\mu = \log(126) - \sigma^2/2$  and simulate the data from the generative model.

Due to the greater computational cost of simulation in this example, we only run the 100 independent chains for 1,600 iterations. The ground truth statistics are calculated from a NUTS chain of  $10^6$  iterations.<sup>4</sup> The path length of RMHMC is uniformly sampled from the range 1 to 30 as done in Girolami and Calderhead (2011).

Performance comparisons as in the previous examples are shown in Fig. 11 and 12. Recycled RMHMC yields worse ESS for the majority of the mean estimators in this example, but this is an artifact of the negative auto-correlations in non-recycled RMHMC causing *super-efficiency* — MCMC samples yielding smaller Monte Carlo

<sup>4</sup>NUTS is actually a meta-algorithm that provides a useful trajectory termination criterion for any MCMC algorithm based on reversible dynamics. In particular, NUTS and our recycled version apply straightforwardly to most of the HMC variants, including RMHMC.

errors than the same number of independent samples. HMC, and hence RMHMC, is known to occasionally exhibit such behavior (Kennedy and Pendleton, 1991; Neal, 2010). We can see, however, that recycling uniformly and significantly improves ESS for the variance and quantile estimators. Under NUTS, such super-efficiency artifacts are not observed and recycling improves ESS for all the estimators. On average, 15 samples were recycled for RMHMC and 7 out of  $2^4 = 16$  for NUTS.

## 5.6 Number of recycled samples and statistical efficiency

As mentioned earlier, in the simulation results above we recycle enough of the intermediate states to achieve near-optimal efficiency gains. Here we take a closer look at how the efficiency gain from recycling depends on the number of recycled samples. The results in particular provide a practical guidance on how one might trade off statistical efficiency for memory efficiency when the available memory becomes limited.

For our experiments here, we focus on the problem of estimating a quantile; the dependence of mean and variance estimators on the number of recycled samples was found to be similar. The number of samples per iteration was repeatedly reduced by a factor of 2 until the benefit of recycling became almost negligible. The results are summarized in the  $\log_2$  ESS ratio plots as presented earlier; Figure 13 for the multivariate Gaussian example, Figure 14 for the hierarchical Bayesian logistic regression example, Figure 15 for the stochastic volatility example, and Figure 16 for the log-Gaussian Cox example. The green dotted line corresponds to the number of recycled samples at which the efficiency decrease relative to the optimal one becomes visually noticeable. The cyan dashed line corresponds to the number of recycled samples below which the benefit from recycling becomes negligible.

The performance of recycled NUTS is particularly remarkable, not only offering the near-optimal efficiency gain well-below the maximal recycling size but also demonstrating over 40% ( $\approx 2^{0.5}$ ) efficiency gain with just one recycled sample. For recycled HMC,

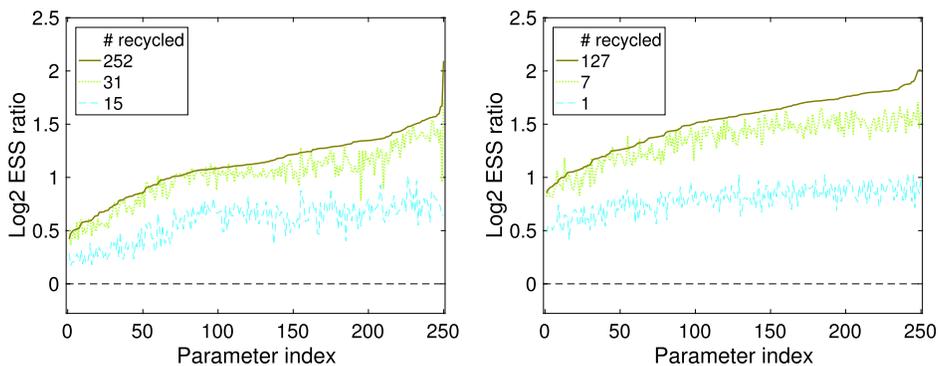


Figure 13: Multivariate Gaussian example: improvement in ESS for 97.5% quantile estimation with different number of recycled samples.

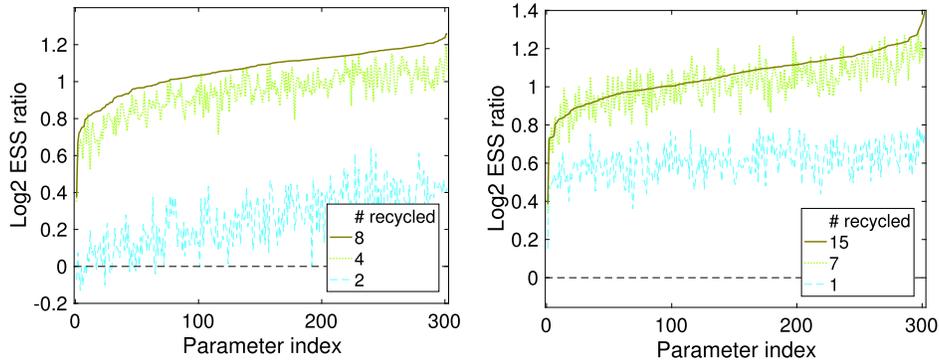


Figure 14: Hierarchical logistic example: improvement in ESS for 97.5% quantile estimation with different number of recycled samples.

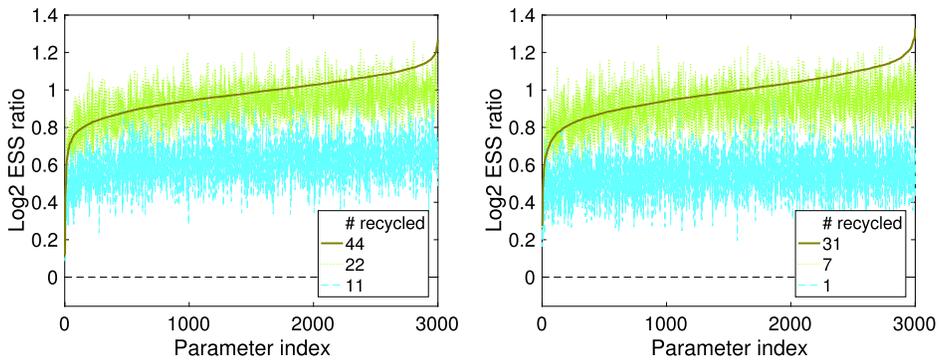


Figure 15: Stochastic volatility example: improvement in ESS for 97.5% quantile estimation with different number of recycled samples.

the efficiency gains remain substantial well-below the maximal recycling size but start to diminish much earlier than NUTS. Two design features of NUTS likely explain this phenomenon. First, NUTS simulates a trajectory in both the forward and backward direction, which means that some of the intermediate states lie in the direction opposite to the final proposal state relative to the starting point of a trajectory. Secondly, while HMC simulates a trajectory to construct one high-quality proposal state, NUTS generates a collection of states — any of which likely constitutes a good proposal state — and selects one state from the collection as a final proposal. Compared to those of HMC, therefore, the recyclable states of NUTS probably have smaller correlations with the final proposal state.

Our experiments here suggest that recycled NUTS may be a particularly practical alternative to the standard implementation of HMC-type algorithms; it not only elimi-

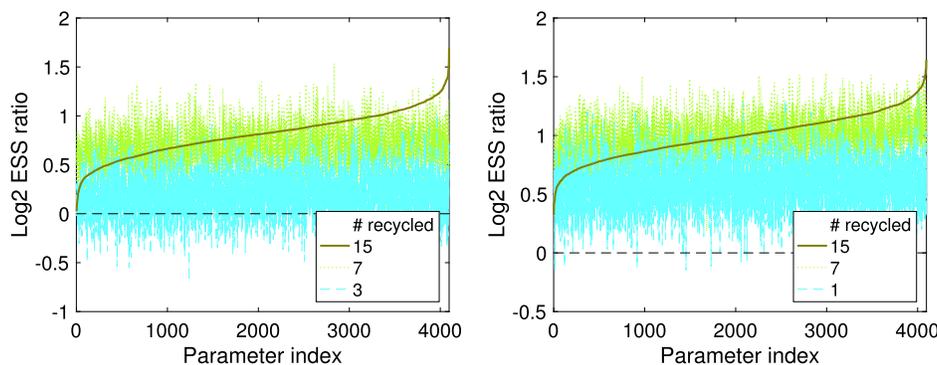


Figure 16: Log-Gaussian Cox example: improvement in ESS for 97.5% quantile estimation with different number of recycled samples.

notes the need to tune the path length but also provides a significant boost in efficiency with a rather small increase in memory requirement.

## 6 Discussion

We have proposed a simple and general algorithm for improving the efficiency of HMC and variants with essentially no extra computational overhead. Our simulations demonstrate the substantial gains in computational efficiency without excessive memory use. In practice, conceptual complexity, ease of implementation, and memory efficiency are just as important considerations as statistical efficiency. These considerations can explain why related ideas to improve the efficiency of HMC variants have not gained traction. Our algorithm provides a more practical and user-friendly alternative that applies straightforwardly to a wide range of multi-proposal schemes.

## Supplementary Material

Supplementary Material for ‘Recycling Intermediate Steps to Improve Hamiltonian Monte Carlo’ (DOI: [10.1214/19-BA1171SUPP](https://doi.org/10.1214/19-BA1171SUPP); .pdf).

## References

- Andrieu, C. and Thoms, J. (2008). “A tutorial on adaptive MCMC.” *Statistics and Computing*, 18(4): 343–373. MR2461882. doi: <https://doi.org/10.1007/s11222-008-9110-y>. 1088
- Bernton, E., Yang, S., Chen, Y., Shephard, N., and Liu, J. S. (2015). “Locally weighted Markov chain Monte Carlo.” *ArXiv e-prints:1506.08852*. 1088, 1092

- Beskos, A., Pillai, N., Roberts, G., Sanz-Serna, J.-M., and Stuart, A. (2013). “Optimal tuning of the hybrid Monte Carlo algorithm.” *Bernoulli*, 19(5A): 1501–1534. MR3129023. doi: <https://doi.org/10.3150/12-BEJ414>. 1087, 1094
- Brooks, S., Gelman, A., Jones, G., and Meng, X.-L. (eds.) (2011). *Handbook of Markov Chain Monte Carlo*. Chapman and Hall/CRC. MR2742422. doi: <https://doi.org/10.1201/b10905>. 1094
- Calderhead, B. (2014). “A general construction for parallelizing Metropolis-Hastings algorithms.” *Proceedings of the National Academy of Sciences*, 111(49): 17408–17413. 1088, 1092
- Duane, S., Kennedy, A., Pendleton, B. J., and Roweth, D. (1987). “Hybrid Monte Carlo.” *Physics Letters B*, 195(2): 216 – 222. MR3960671. 1087
- Fang, Y., Sanz-Serna, J. M., and Skeel, R. D. (2014). “Compressible generalized hybrid Monte Carlo.” *The Journal of Chemical Physics*, 140(17). 1088, 1092
- Frenkel, D. (2004). “Speed-up of Monte Carlo simulations by sampling of rejected states.” *Proceedings of the National Academy of Sciences of the United States of America*, 101(51): 17571–17575. 1088
- Gelman, A., Carlin, J. B., Stern, H. S., Dunson, D. B., Vehtari, A., and Rubin, D. B. (2013). *Bayesian Data Analysis*. CRC Press. MR3235677. 1087
- Girolami, M. and Calderhead, B. (2011). “Riemann manifold Langevin and Hamiltonian Monte Carlo methods.” *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(2): 123–214. MR2814492. doi: <https://doi.org/10.1111/j.1467-9868.2010.00765.x>. 1088, 1092, 1094, 1101, 1102
- Hastie, D. I. and Green, P. J. (2012). “Model choice using reversible jump Markov chain Monte Carlo.” *Statistica Neerlandica*, 66(3): 309–338. MR2955422. doi: <https://doi.org/10.1111/j.1467-9574.2012.00516.x>. 1092
- Hoffman, M. D. and Gelman, A. (2014). “The No-U-turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo.” *J. Mach. Learn. Res.*, 15(1): 1593–1623. MR3214779. 1088, 1090, 1092, 1093, 1094, 1099
- Kallenberg, O. (2002). *Foundations of Modern Probability*. Probability and Its Applications. Springer New York. MR1876169. doi: <https://doi.org/10.1007/978-1-4757-4015-8>. 1091
- Kennedy, A. and Pendleton, B. (1991). “Acceptances and autocorrelations in hybrid Monte Carlo.” *Nuclear Physics B-Proceedings Supplements*, 20: 118–121. 1103
- Kruschke, J. (2014). *Doing Bayesian Data Analysis, Second Edition: A Tutorial with R, JAGS, and Stan*. Academic Press. 1087
- Lan, S., Zhou, B., and Shahbaba, B. (2014). “Spherical Hamiltonian Monte Carlo for Constrained Target Distributions.” In *Proceedings of the 31st International Conference on Machine Learning, ICML’14*, 629–637. 1088

- Livingstone, S., Betancourt, M., Byrne, S., and Girolami, M. (2016). “On the Geometric Ergodicity of Hamiltonian Monte Carlo.” *arXiv:1601.08057*. MR3648031. doi: <https://doi.org/10.3150/16-BEJ810>. 1094
- Livingstone, S., Faulkner, M. F., and Roberts, G. O. (2017). “Kinetic energy choice in Hamiltonian/hybrid Monte Carlo.” *arXiv:1706.02649*. MR3949305. doi: <https://doi.org/10.1093/biomet/asz013>. 1094
- Lu, X., Perrone, V., Hasenclever, L., Teh, Y. W., and Vollmer, S. J. (2016). “Relativistic Monte Carlo.” *ArXiv e-prints: 1609.04388*. 1088
- Mangoubi, O. and Smith, A. (2017). “Rapid mixing of Hamiltonian Monte Carlo on strongly log-concave distributions.” *arXiv preprint arXiv:1708.07114*. 1094
- Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E. (1953). “Equation of State Calculations by Fast Computing Machines.” *The Journal of Chemical Physics*, 21(6): 1087–1092. 1088
- Monnahan, C. C., Thorson, J. T., and Branch, T. A. (2016). “Faster estimation of Bayesian models in ecology using Hamiltonian Monte Carlo.” *Methods in Ecology and Evolution*, 339–348. 1087
- Neal, R. M. (1994). “An Improved Acceptance Procedure for the Hybrid Monte Carlo Algorithm.” *Journal of Computational Physics*, 111(1): 194–203. MR1271540. doi: <https://doi.org/10.1006/jcph.1994.1054>. 1088, 1089
- Neal, R. M. (1996). *Bayesian Learning for Neural Networks*. Springer-Verlag. 1087
- Neal, R. M. (2010). “MCMC using Hamiltonian Dynamics.” In *Handbook of Markov Chain Monte Carlo*. CRC Press. MR3185067. 1087, 1088, 1090, 1092, 1094, 1095, 1096, 1103
- Nishimura, A. and Dunson, D. (2019) “Supplementary material for ‘Recycling Intermediate Steps to Improve Hamiltonian Monte Carlo’.” *Bayesian Analysis*. doi: <https://doi.org/10.1214/19-BA1171SUPP>. 1092
- Pakman, A. and Paninski, L. (2013). “Auxiliary-variable Exact Hamiltonian Monte Carlo Samplers for Binary Distributions.” In *Proceedings of the 26th International Conference on Neural Information Processing Systems, NIPS’13*, 2490–2498. 1088
- Pakman, A. and Paninski, L. (2014). “Exact Hamiltonian Monte Carlo for Truncated Multivariate Gaussians.” *Journal of Computational and Graphical Statistics*, 23(2): 518–542. MR3215823. doi: <https://doi.org/10.1080/10618600.2013.788448>. 1088
- Salvatier, J., Wiecki, T. V., and Fonnesbeck, C. (2016). “Probabilistic programming in Python using PyMC3.” *PeerJ Computer Science*. 1087
- Shahbaba, B., Lan, S., Johnson, W. O., and Neal, R. M. (2014). “Split Hamiltonian Monte Carlo.” *Statistics and Computing*, 24(3): 339–349. MR3192260. doi: <https://doi.org/10.1007/s11222-012-9373-1>. 1088, 1092

- Stan Development Team (2015). *Stan Modeling Language Users Guide and Reference Manual, Version 2.9.0*. [1087](#), [1088](#), [1095](#), [1099](#)
- Tjelmeland, H. (2004). “Using all Metropolis-Hastings proposals to estimate mean values.” Technical report, Norwegian University of Science and Technology. [1088](#)
- Wang, Z., Mohamed, S., and de Freitas, N. (2013). “Adaptive Hamiltonian and Riemann manifold Monte Carlo samplers.” In *Proceedings of the 30th International Conference on Machine Learning*, volume 28, 1462–1470. [1088](#), [1100](#)
- Zhang, Y., Wang, X., Chen, C., Heno, R., Fan, K., and Carin, L. (2016). “Towards Unifying Hamiltonian Monte Carlo and Slice Sampling.” *ArXiv:1602.07800*. [1088](#)