# Entropy-based adaptive Hamiltonian Monte Carlo

**Anonymous Author(s)**
Affiliation
Address
email

## Abstract

Hamiltonian Monte Carlo (HMC) is a popular Markov Chain Monte Carlo (MCMC) algorithm to sample from an unnormalized probability distribution. We develop a gradient-based algorithm that allows for the adaptation of the mass matrix by encouraging the leapfrog integrator to have small energy errors while also exploring all dimensions jointly. In contrast to previous work that adapt the hyperparameters of HMC using some form of expected squared jumping distance, the adaptation strategy suggested here aims to increase sampling efficiency by maximizing an approximation of the proposal entropy. We illustrate that using multiple gradients in the HMC proposal can be beneficial compared to a single gradient-step in Metropolis-adjusted Langevin proposals, although such approximations are not necessary for computing the proposal entropy with a single leapfrog step. Empirical evidence suggests that the adaptation method can outperform different versions of HMC schemes by adjusting the mass matrix to the geometry of the target distribution and by providing some control on the integration time.

## 1 Introduction

Consider the problem of sampling from a target density $\pi$ on $\mathbb{R}^d$ of the form $\pi(q) \propto \mathrm{e}^{-U(q)}$, with a *potential* energy $U\colon \mathbb{R}^d \to \mathbb{R}$ being twice continuously differentiable. HMC methods [14, 39, 6] sample from a *Boltzmann-Gibbs* distribution $\mu(q,p) \propto \mathrm{e}^{-H(q,p)}$ on the phase-space $\mathbb{R}^{2d}$ based on the (separable) *Hamiltonian* function

$$H(q,p) = U(q) + K(p) \quad \text{with} \quad K(p) = \frac{1}{2} p^\top M^{-1} p.$$

The Hamiltonian represents the total energy that is split into a potential energy term $U$ and a *kinetic* energy $K$ which we assume is Gaussian for some symmetric positive definite *mass matrix* $M$. Suppose that $(q(t), p(t))_{t\in\mathbb{R}}$ evolve according to the differential equations

$$\frac{\mathrm{d}q(t)}{\mathrm{d}t} = \frac{\partial H(q(t),p(t))}{\partial p} = M^{-1}p(t) \quad \text{and} \quad \frac{\mathrm{d}p(t)}{\mathrm{d}t} = -\frac{\partial H(q(t),p(t))}{\partial q} = -\nabla U(q(t)). \tag{1}$$

Let $(\varphi_t)_{t\geqslant 0}$ denote the flow of the Hamiltonian system, that is for fixed $t$, $\varphi_t$ maps each $(q,p)$ to the solution of (1) that takes value $(q,p)$ at time $t = 0$. The exact HMC flow $\varphi$ preserves volume and conserves the total energy *i.e.* $H \circ \varphi_t = H$. Consequently, the Boltzmann-Gibbs distribution $\mu$ is invariant under the Hamiltonian flow, that is $\mu(\varphi_t(E)) = \mu(E)$ for any Borel set $E \subset \mathbb{R}^{2d}$. Furthermore, the flow satisfies the generalized *reversibility* condition $\mathcal{F} \circ \varphi_t = \varphi_{-t} \circ \mathcal{F}$ with the flip operator $\mathcal{F}(q,p) = (q,-p)$. Put differently, the Hamiltonian dynamics go backward in time by negating the velocity. If an analytical expression for the exact flow were available, one could sample from $\mu$ using the invariant Markov chain that at state $(q,p)$ first draws a new velocity $p' \sim \mathcal{N}(0,M)$ with the next state set to $\varphi_T(q,p')$ for some *integration time* $T > 0$. Such a velocity refreshment is necessary as the HMC dynamics preserve the energy and so cannot

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

be ergodic. However, the Hamiltonian flow cannot be computed exactly, except for very special potential functions. Numerical approximations to the exact solution of Hamiltonian's equations are thus routinely used, most commonly the *leapfrog* method, also known as (velocity) Verlet integrator [21, 7]. For a step size $h > 0$ and $L$ steps, such an algorithm updates the previous state $q_0$ and a new velocity $p_0 \sim \mathcal{N}(0, M)$ by setting, for $0 \leqslant \ell \leqslant L-1$,

$$p_{\ell+\frac{1}{2}} = p_\ell - \frac{h}{2}\nabla U(q_\ell); \quad q_{\ell+1} = q_\ell + hM^{-1}p_{\ell+\frac{1}{2}}; \quad p_{\ell+1} = p_{\ell+\frac{1}{2}} - \frac{h}{2}\nabla U(q_{\ell+1}).$$

This scheme can be motivated by splitting the Hamiltonian wherein the kick mappings in the first and third step update only the momentum, while the drift mapping in the second step advances only the position $q$ with constant speed. For $T = Lh$, the leapfrog integrator approximates $\varphi_T(q_0, p_0)$ by $(q_L, p_L)$ while also preserving some geometric properties of $\varphi$, namely volume preservation and generalized reversibility. The leapfrog method is a second-order integrator, making an $\mathcal{O}(h^2)$ energy error $H(q_L, p_L) - H(q_0, p_0)$. A $\mu$-invariant Markov chain can be constructed using a Metropolis-Hastings acceptance step. More concretely, the proposed state $(q_L, p_L)$ is accepted with the acceptance rate $a(q_0, p_0) = \min\{1, \exp[-(H(q_L, p_L) - H(q_0, p_0))]\}$, while the next state is set to $\mathcal{F}(q_0, p_0)$ in case of rejection, although the velocity flip is inconsequential for full refreshment strategies.

## 2 Related work

The choice of the hyperparameters $h$, $L$ and $M$ can have a large impact on the efficiency of the sampler. For fixed $L$ and $M$, a popular approach for adapting $h$ is to target an acceptance rate of around $0.65$ which is optimal for iid Gaussian targets in the limit $d \to \infty$ [5]. HMC hyperparameters have been tuned using some form of *expected squared jumping distance* (ESJD) [41], using for instance Bayesian optimization [47] or a gradient-based approach [33]. A popular approach suggested in [25] tunes $L$ based on the ESJD by doubling $L$ until the path makes a U-turn and retraces back towards the starting point, that is by stopping to increase $L$ when the distance to the proposed state reaches a stationary point [1]; see also [48] for a variation and [40] for a version using sequential proposals. The Riemann manifold HMC algorithm from [18] has been suggested that uses a position dependent mass matrix $M(x)$ based on a non-separable Hamiltonian, but can be computationally expensive, requiring $\mathcal{O}(d^3)$ operations in general. An alternative to choose $M$ or more generally the kinetic energy $K$ was proposed in [36] by analysing the behaviour of $x \mapsto \nabla K(\nabla U(x))$. Different preconditioning approaches have been compared for Gaussian targets in [31]. A popular route has also been to first transform the target using tools from variational inference as in [24] and then run a HMC sampler with unit mass matrix on the transformed density with a more favourable geometry.

A common setting to study the convergence of HMC assumes a log-concave target. In the case that $U$ is $m_1$-strongly convex and $m_2$-smooth, [38, 10] analyse the ideal HMC algorithm with unit mass matrix where a higher condition number $\kappa = m_2/m_1$ implies slower mixing: The relaxation time, *i.e.* the inverse of the spectral gap, grows linear in $\kappa$, assuming the integration time is set to $T = \frac{1}{2\sqrt{m_2}}$. [9] establish non-asymptotic upper bounds on the mixing time using a leap-frog integrator where the step size $h$ and the number $L$ of steps depends explicitly on $m_1$ and $m_2$. Convergence guarantees are established using conductance profiles by obtaining (i) a high probability lower bound on the acceptance rate and (ii) a lower bound on the KL-divergence between the HMC proposal densities at the starting positions $q_0$ and $q_0'$, whenever $q_0$ is close to $q_0'$.

We want to explore here further the generalised speed measure introduced in [45] for adapting RWM or MALA that aim to achieve fast convergence by constructing proposals that (i) have a high average log-acceptance rate and (ii) have a high entropy. Whereas the entropy of the proposal in RWM or MALA algorithms can be evaluated efficiently, the multi-step nature of the HMC trajectories makes this computation less tractable. The recent work in [34] consider the same adaptation objective by learning a normalising flow that is inspired by a leapfrog proposal with a more tractable entropy by masking components in a leapfrog-style update via an affine coupling layer as used for RealNVPs [13]. [49] sets the integration time by maximizing the proposal entropy for the exact HMC flow in Gaussian targets, while choosing the mass matrix to be the inverse of the sample covariance matrix.

# 3 Entropy-based adaptation scheme

## 3.1 Marginal proposal entropy

Suppose that $CC^\top = M^{-1}$, where $C$ is defined by some parameters $\theta$ and can be a diagonal matrix, a full Cholesky factor etc. Without loss of generality, the step size $h > 0$ can be fixed. We can reparameterise the momentum resampling step $p_0 \sim \mathcal{N}(0, M)$ by sampling $v \sim \mathcal{N}(0, \mathrm{I})$ and setting $p_0 = C^{-\top} v$. One can show by induction that the $L$-th step position $q_L$ and momentum $p_L$ of the leapfrog integrator can be represented as a function of $v$ via

$$q_L = \mathcal{T}_L(v) = q_0 - \frac{Lh^2}{2} M^{-1} \nabla U(q_0) + LhCv - h^2 M^{-1} \Xi_L(v), \tag{2}$$

and

$$p_L = \mathcal{W}_L(v) = C^{-\top} v - \frac{h}{2} \left[ \nabla U(q_0) + \nabla U \circ \mathcal{T}_L(v) \right] - h \sum_{i=1}^{n-1} \nabla U \circ \mathcal{T}_i(v)$$

where

$$\Xi_L(v) = \sum_{i=1}^{L-1} (L - i) \nabla U \circ \mathcal{T}_L(v), \tag{3}$$

see also [35, 15, 9] for the special case with an identity mass matrix. Observe that for $L = 1$ leap-frog steps, this reduces to a MALA proposal with preconditioning matrix $M^{-1}$.

Under regularity conditions, see for instance [15], the transformation $\mathcal{T}_L \colon \mathbb{R}^d \to \mathbb{R}^d$ is a $\mathrm{C}^1$-diffeomorphism. With $\nu$ denoting the standard Gaussian density, the density $r_L$ of the HMC proposal for the position $q_L$ after $L$ leapfrog steps is the pushforward density of $\nu$ via the map $\mathcal{T}_L$ so that[1]

$$\log r_L(\mathcal{T}_L(v)) = \log \nu(v) - \log |\det \mathsf{D}\mathcal{T}_L(v)|. \tag{4}$$

Observe that the density depends on the Jacobian of the transformation $\mathcal{T}_L \colon v \mapsto q_L$. We would like to avoid computing $\log |\det \mathsf{D}\mathcal{T}_L(v)|$ exactly. Define the residual transformation

$$\mathcal{S}_L \colon \mathbb{R}^d \to \mathbb{R}^d, \; v \mapsto \frac{1}{Lh} C^{-1} \mathcal{T}_L(v) - v. \tag{5}$$

Then $\mathsf{D}\mathcal{T}_L(v) = LhC(\mathrm{I} + \mathsf{D}\mathcal{S}_L(v))$ and consequently

$$\log |\det \mathsf{D}\mathcal{T}_L(v)| = d \log(Lh) + \log |\det C| + \log |\det(\mathrm{I} + \mathsf{D}\mathcal{S}_L(v))|. \tag{6}$$

Combining (4) and (6) yields the log-probability of the HMC proposal

$$\log r_L(\mathcal{T}_L(v)) = \log \nu(v) - d \log(Lh) - \log |\det C| - \log |\det(\mathrm{I} + \mathsf{D}\mathcal{S}_L(v))|. \tag{7}$$

Comparing the equations (2) and (5), one sees that $\mathcal{S}_L(v) = c - \frac{h}{L} C^\top \Xi_L(v)$ for some constant $c \in \mathbb{R}^d$ that depends on $\theta$ but is independent of $v$ and consequently, $\mathsf{D}\mathcal{S}_L(v) = -\frac{h}{L} C^\top \mathsf{D}\Xi_L(v)$. We next show a recursive expression for $\mathsf{D}\mathcal{S}_L$ with a proof given in Appendix B.

**Lemma 1** (Jacobian representation)**.** *It holds that* $\mathsf{D}\mathcal{S}_1 = 0$ *and for any* $\ell \in \{2, \dots, L\}$, $v \in \mathbb{R}^d$,

$$\mathsf{D}\mathcal{S}_\ell(v) = -h^2 \sum_{i=1}^{\ell-1} (\ell - i) \frac{i}{\ell} C^\top \nabla^2 U\left(\mathcal{T}_i(v)\right) C \left(\mathrm{I} + \mathsf{D}\mathcal{S}_i(v)\right). \tag{8}$$

*In particular,* $\mathsf{D}\mathcal{S}_\ell(v)$ *is a symmetric matrix. Suppose further that* $L^2 h^2 < \sup_{q \in \mathbb{R}^d} \frac{1}{4 \| C^\top \nabla^2 U(q) C \|_p}$. *Then for any* $\ell \in \{1, \dots, L\}$ *and* $v \in \mathbb{R}^d$, *we have* $\|\mathsf{D}\mathcal{S}_\ell(v)\|_p < \frac{1}{8}$.

Consider for the moment a Gaussian target with potential function $U(q) = \frac{1}{2}(q - q_\star)^\top \Sigma^{-1} (q - q_\star)$ for $q_\star \in \mathbb{R}^d$ and positive definite $\Sigma \in \mathbb{R}^{d \times d}$. Then, due to (8), for any $q \in \mathbb{R}^d$, $v \in \mathbb{R}^d$,

$$\mathsf{D}\mathcal{S}_L(v) = -h^2 \sum_{i=1}^{L-1} (L - i) \frac{i}{L} C^\top \Sigma^{-1} C (\mathrm{I} + \mathsf{D}\mathcal{S}_i(v)) = D_L + R_L(v),$$

[1] We denote the Jacobian matrix of a function $f \colon \mathbb{R}^d \to \mathbb{R}^d$ at the point $x$ as $\mathsf{D}f(x)$.

where

$$D_L = -h^2 C^\top \Sigma^{-1} C \left( \sum_{i=1}^{L-1} (L-i) \frac{i}{L} \right) = -h^2 \frac{L^2-1}{6} C^\top \Sigma^{-1} C$$

and a remainder term $R_L(v) = -h^2 C^\top \Sigma^{-1} C \left( \sum_{i=1}^{L-1} (L-i) \frac{i}{L} \mathsf{D}\mathcal{S}_i(v) \right)$. From Lemma 1, we see that if $\left\| C^\top \Sigma^{-1} C \right\|_2 \leqslant \frac{h^2}{4L^2}$, then $\mathrm{I} + \mathsf{D}\mathcal{S}_L(v)$ and $-\mathsf{D}\mathcal{S}_L(v)$ are positive definite. Then $R_L$ is also positive definite and $\log\det(\mathrm{I}+D_L) \leqslant \log|\det(\mathrm{I}+\mathsf{D}\mathcal{S}_L(v))|$ and we can maximize the lower bound instead. Notice that $R_2 = 0$ and one can include higher order terms $\mathcal{O}\left( \left[ h^2 C^\top \Sigma^{-1} C \right]^k \right)$, $k > 1$, in the approximation $D_L$, but we have not explored this systematically.

For an arbitrary potential energy $U$, we suggest to maximize

$$\mathcal{L}(\theta) = \log|\det(\mathrm{I}+D_L)| \quad \text{with} \quad D_L = -h^2 \frac{L^2-1}{6} C^\top \nabla^2 U(q_{\lfloor L/2 \rfloor}) C \tag{9}$$

as an approximation of $\log|\det(\mathrm{I}+\mathsf{D}\mathcal{S}_L)|$. The intuition is that we assume that the target density can be approximated locally by a Gaussian one with precision matrix given by the Hessian of $U$ at the mid-point $q_{\lfloor L/2 \rfloor}$ of the trajectory. Unlike in the Gaussian case, the matrix $D_L$ depends on $q_0$ and $v$, although we make this dependence not explicit to simplify the notation. In general, however, neither $\mathsf{D}\mathcal{S}_L(v)$, $\mathrm{I}+\mathsf{D}\mathcal{S}_L(v)$, $R_L(v)$ nor $D_L$ need to be positive definite. We want to optimize $\mathcal{L}(\theta)$ given in (9) even if we do not have access to the Hessian $\nabla^2 U$ explicitly, but only through Hessian-vector products $\nabla^2 U(q) w$ for some vector $w \in \mathbb{R}^d$. Vector-Jacobian products $\mathtt{vjp}(f, x, w) = w^\top \mathsf{D} f(x)$ for differentiable $f \colon \mathbb{R}^d \to \mathbb{R}^d$ can be computed efficiently via reverse-mode automatic differentiation, so that $\nabla^2 U(q) w = \mathtt{vjp}(\nabla U, q, w)^\top$ can be evaluated with complexity linear in $d$.

Suppose the multiplication with $D_L$ is a contraction so that all eigenvalues of $D_L$ have absolute values smaller than one. Then one can apply a Hutchinson stochastic trace estimator of $\log|\det(\mathrm{I}_d + D_{,L})|$ with a Taylor approximation, truncated and re-weighted using a Russian-roulette estimator [37], see also [22, 2, 8] for similar approaches in different settings. More concretely, let $N$ be a positive random variable with support on $\mathbb{N}$ and let $p_k = \mathbb{P}(N \geqslant k)$. Then,

$$\mathcal{L}(\theta) = \log\det(\mathrm{I}+D_L) = \mathbb{E}_{N,\varepsilon} \left[ \sum_{k=1}^{N} \frac{(-1)^{k+1}}{k p_k} \varepsilon^\top (D_L)^k \varepsilon \right], \tag{10}$$

where $\varepsilon$ is drawn from a Rademacher distribution. While this yields an unbiased estimator for $\mathcal{L}(\theta)$ and its gradient as shown in Appendix A.1 if $D_L$ is contractive, it can be computationally expensive if $N$ has a large mean or have a high variance if $D_L$ has an eigenvalue that is close to 1 or $-1$, see [37, 12]. Since both the first order Gaussian approximation as well as the Russian Roulette estimator hinges on $D_L$ having small absolute eigenvalues, we consider a constrained optimisation approach that penalises such large eigenvalues. For the random variable $N$ that determines the truncation level in the Taylor series, we compute $b_N = (D_L)^N \varepsilon / \left\| (D_L)^N \varepsilon \right\|_2$ and $\mu_N = b_N^\top D_L b_N$. Note that this corresponds to applying $N$ times the power iteration algorithm and with $|\lambda_1| > |\lambda_2| \geqslant \ldots \geqslant |\lambda_d|$ denoting the eigenvalues of the symmetric matrix $D_L$, almost surely $\mu_n \to \lambda_1$ for $n \to \infty$, see [19]. For some $\delta \in (0,1)$, we choose some differentiable monotone increasing penalty function $h \colon \mathbb{R} \to \mathbb{R}$ such that $h(x) > 0$ for $x > \delta$ and $h(x) = 0$ for $x \leqslant \delta$ and we add the term $\gamma h(|\mu_N|)$ for $\gamma > 0$ to the loss function that we introduce below, see Appendix A.2 for an example of $h$.

### 3.2 Adaptation with a generalised speed measure

Extending the objective from [45] to adapt the HMC proposal, we aim to solve

$$\arg\min_\theta \int \int \pi(q_0)\nu(v) \Big[ -\log a\left((q_0, v), (\mathcal{T}_L(v), \mathcal{W}_L(v))\right) + \beta \log r_L(\mathcal{T}_L(v)) \Big] \mathrm{d}v \mathrm{d}q_0, \tag{11}$$

where $\mathcal{T}_L$, $\mathcal{W}_L$, $r_L$ as well as the acceptance rate $a$ depend on $q_0$ and the parameters $\theta$ we want to adapt. Also, the hyper-parameter $\beta > 0$ can be adapted online by increasing $\beta$ if the acceptance rate is above a target acceptance rate $\alpha_\star$ and decreasing $\beta$ otherwise. We choose $\alpha_\star = 0.67$, which is optimal for increasing $d$ under independence assumptions [5]. One part of the objective constitutes minimizing the energy error $\Delta(q_0, v) = H(\mathcal{T}_L(v), \mathcal{W}_L(v)) - H(q_0, C^{-\top} v)$ that determines the

log-acceptance rate via $\log a(q_0, C^{-\top}v) = \min\{0, -\Delta(q_0, v)\}$. Unbiased gradients of the energy error can be obtained without stopping any gradient calculations in the backward pass. However, we found that a multi-step extension of the biased fast MALA approximation from [45] tends to improve the adaptation by stopping gradients through $\nabla U$ as shown in Appendix A.3.

Suppose that the current state of the Markov chain is $q$. We resample the momentum $v \sim \mathcal{N}(0, \mathrm{I})$ and aim to solve (11) by taking gradients of the penalised loss function

$$-\min\{0, -\Delta(q, v)\} - \beta\left(d \log h + \log|\det C| + \mathcal{L}(\theta) - \gamma h(|\mu_N|)\right),$$

as illustrated in Algorithm 1, which also shows how we update the hyperparameters $\beta$ and $\gamma$. Algorithm 1 requires to choose learning rates $\rho_\theta$, $\rho_\beta$ and $\rho_\gamma$. Instead of simulating a single Markov chain, the parameters can be updated using multiple parallel chains. We used 10 parallel chains throughout our experiments. The adaptive algorithm should also allow to make advantage of SIMD (single instruction, multiple data) operations, thereby benefiting from hardware accelerators such as GPUs. Pseudo-code for simulating from the adaptive chain can found in Algorithm 1. Our implementation builds up on tensorflow probability [32] with some target densities taken from [44].

**Algorithm 1** Sample the next state $q'$ and adapt $\beta$, $\gamma$ and $\theta$.

Sample velocity $v \sim \mathcal{N}(0, \mathrm{I})$ and set $p = C^{-\top}v$.
Apply integrator $\mathtt{LF}$ to obtain $(q_\ell, p_\ell, \nabla U(q_\ell))_{0 \leqslant \ell \leqslant L} = \mathtt{LF}(q, p)$.
Stop gradients $\nabla U(q_\ell) = \mathtt{stop_grad}(\nabla U(q_\ell))$ for $0 \leqslant \ell \leqslant L$.
Compute $\Xi_L(v)$ using (3).
Compute $\Delta(q_0, v)$ using (13) and set $a = \min\{1, \mathrm{e}^{-\Delta(q_0, v)}\}$.
Compute $\bar{\eta}_N, y = \textsc{Rademacher}()$.
Set $\mathcal{L}(\theta) = \mathtt{stop_grad}(y)^\top D_L \varepsilon$.
Set $b_N = \mathtt{stop_grad}\left(\frac{\bar{\eta}_N}{\|\bar{\eta}_N\|_2^2}\right)$ and $\mu_N = b_N^\top D_L b_N$.
$\mathcal{E}(\theta) = -\min\{0, -\Delta(q_0, v)\} - \beta\left(d \log h + \log|\det C| + \mathcal{L}(\theta) - \gamma h(|\mu_N|)\right)$.
Adapt $\theta \leftarrow \theta - \rho_\theta \nabla_\theta \mathcal{E}(\theta)$.
Adapt $\beta \leftarrow \Pi_\beta\left[\beta(1 + \rho_\beta(a - \alpha_\star))\right]$. #$\Pi_\beta$ projects onto a compact set; default value $[10^{-2}, 10^2]$.
Adapt $\gamma \leftarrow \Pi_\gamma\left[\gamma + \rho_\gamma h(|\mu_N|)\right]$. #$\Pi_\gamma$ projects onto a compact set; default value $[10^3, 10^5]$.
Sample $u \sim \mathcal{U}(0, 1)$ and set $q' = \mathbf{1}_{\{u \leqslant a\}}\, q_L + \mathbf{1}_{\{u > a\}}\, q$.

**function** $D_L(w)$:
  #$D_L(w) = D_L w$ computes Hessian-vector products efficiently
  $z = \mathtt{vjp}(\nabla U, \mathtt{stop_grad}(q_{\lfloor L/2 \rfloor}), Cw)^\top$
  **return** $-h^2 \frac{L^2-1}{6}\, C^\top z$
**end function**

**function** $\textsc{Rademacher}$:
  Sample Rademacher random variable $\varepsilon$ and truncation level $N$.
  Initialise $y \leftarrow 0$ and $\bar{\eta}_0 = \varepsilon$.
  **for** $k = 1 \ldots N$ **do**
    #Apply a spectral normalisation for stability if $D_L$ is not a contraction; $\delta' \in (0, 1)$.
    Set $\bar{\eta}_k = D_L \bar{\eta}_{k-1} \cdot \min\left\{1, \delta' \|\bar{\eta}_{k-1}\|_2 / \|D_L \bar{\eta}_{k-1}\|_2\right\}$ and $y \leftarrow y + \frac{(-1)^k}{p_k} \bar{\eta}_k$.
  **end for**
  **return** $\bar{\eta}_N, y$
**end function**

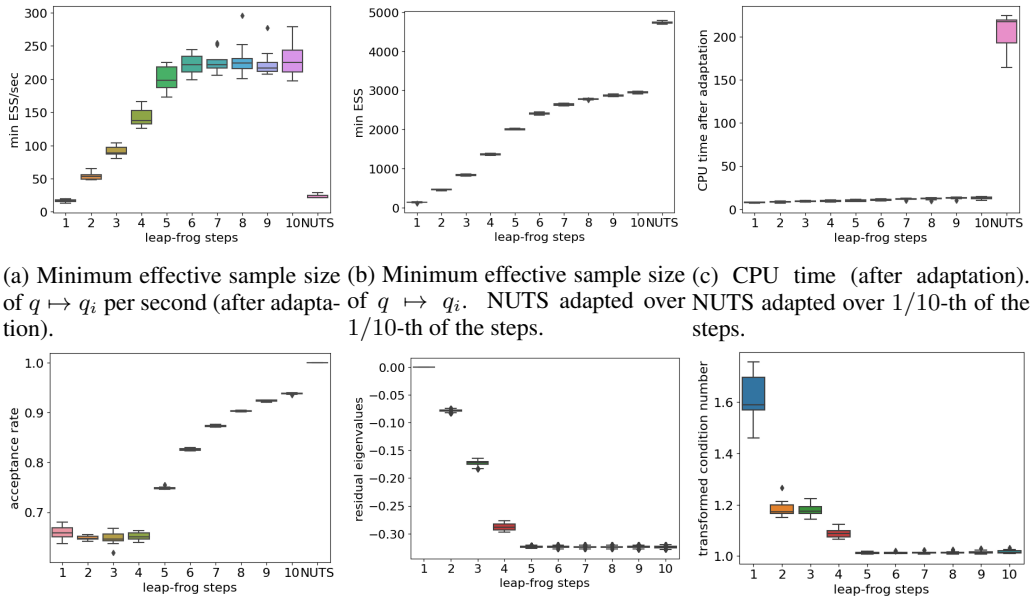
# 4 Numerical experiments

## 4.1 Gaussian targets

**Anisotropic Gaussian distributions.** We consider sampling from a multivariate Gaussian distribution $\mathcal{N}(0, \Sigma)$ with strictly convex potential $U(q) = \frac{1}{2} q^\top \Sigma^{-1} q$ for different covariance matrices $\Sigma$. For $c > 0$, assume a covariance matrix given by $\Sigma_{ij} = \delta_{ij} \exp\left(c(i-1)/(d-1) \log 10\right)$. We set (i) $c = 3$ and $d \in \{10^3, 10^4\}$ and (ii) $c = 6$ and $d = 100$, as considered in [43]. The eigenvalues of the covariance matrix are thus distributed between 1 to 100 in setting (i), while they vary

from 1 and $10^6$ in setting (ii). The preconditioning factor $C$ is assumed to be diagonal. We adapt the sampler for $4 \times 10^4$ steps in case (i) and for $10^5$ steps in case (ii). We compute the minimum effective sample size (minESS) of all functions $q \mapsto q_i$ over $i \in \{1, \dots, d\}$ as shown in Figure 1 for $d = 10^3$ with leapfrog steps ranging from $L = 1$ to 10. We also compared it with a NUTS implementation in tensorflow probability [32] with a default maximum tree depth of 10 and step sizes adapted using dual averaging. HMC performs well in terms of minESS/sec for $L > 1$ and the mass matrix adapts to the target covariance with the condition number of $C^\top \Sigma^{-1} C$ becoming relatively close to 1. The entropy objective also yields acceptance rates approaching 1 for increasing leap-frog steps and multiplication with $D_L$ becomes a contraction. Results for $d = 10^4$ can be found in Figure 7 in Appendix D.1 which indicate that as the dimension increases, using a more leap-frog steps becomes more advantageous. For the case (ii) of a very ill-conditioned target, results from Figure 8 in Appendix D.2 also show that $L > 1$ is beneficial and that NUTS without some adaptation of the mass matrix mixes less efficiently. We want to emphasize that for fixed $L$, high acceptance rates for HMC need not be disadvantageous. This is illustrated in Figure 10 in Appendix D.4 for a Gaussian target $\mathcal{N}(0, \mathrm{I})$ in dimension $d = 10$, where tuning just the step-size to achieve a target acceptance rate can lead to slow mixing for some $L$, because the proposal can make a U-turn.

**Correlated Gaussian distribution.** We sample from a 51-dimensional Gaussian target with covariance matrix given by the squared exponential kernel plus small white noise as in [45], with $k(x_i, x_j) = \exp\left(-\frac{1}{2}(x_i - x_j)^2/0.4^2\right) + .01\delta_{ij}$ on the regular grid $[0, 4]$. We consider a general Cholesky factor $C$. The adaptation is performed over $10^5$ steps. Results over 10 runs are shown in Figure 9 in Appendix D.3 which indicates that HMC with moderate $L$ around 4 performs best.




(a) Minimum effective sample size of $q \mapsto q_i$ per second (after adaptation).

(b) Minimum effective sample size of $q \mapsto q_i$. NUTS adapted over $1/10$-th of the steps.

(c) CPU time (after adaptation). NUTS adapted over $1/10$-th of the steps.

(d) Average acceptance rate (after adaptation).

(e) Estimates of the eigenvalues of $D_L$ using power iteration.

(f) Condition number of transformed Hessian $C^\top \Sigma^{-1} C$.

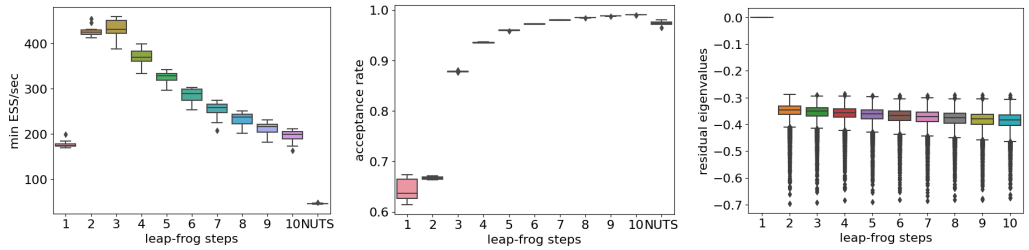
Figure 1: Anisotropic Gaussian target ($d = 1000$).

## 4.2 Logistic regression

Consider a Bayesian logistic regression model with $n$ data points $y_i \in \{0, 1\}$ and $d$-dimensional covariates $x_i \in \mathbb{R}^d$ for $i \in \{1, \dots, n\}$. Assuming a Gaussian prior with covariance matrix $\Sigma_0$ implies a potential function $U(q) = \sum_{i=1}^n \left[-y_i x_i^\top q + \log\left(1 + \mathrm{e}^{x_i^\top q}\right)\right] + \frac{1}{2} q^\top \Sigma_0^{-1} q$.

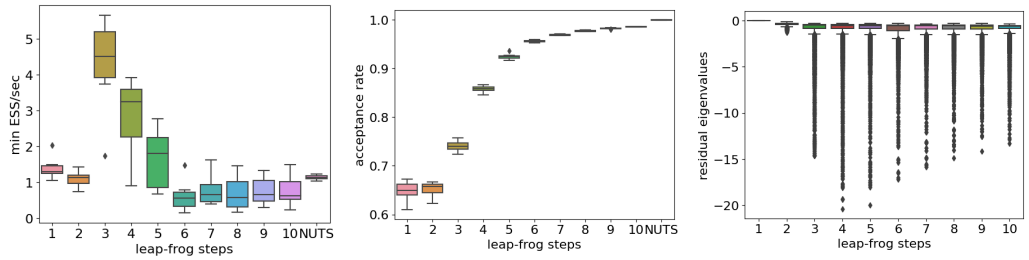
We considered six datasets (Australian Credit, Heart, Pima Indian, Ripley, German Credit and Caravan) that are commonly used for benchmarking inference methods, cf. [11]. The state dimension ranges from $d = 3$ to $d = 87$. We choose $\Sigma_0 = \mathrm{I}$ and parametrize $C$ via a Cholesky matrix. We adapt over $10^4$ steps. HMC with a moderate number of leap-frog steps tends to perform better for

four out of six data sets, with subpar performance for the Australian and Caravan data in terms of minESS/sec, albeit with higher median ESS/sec across dimensions or for computing the potential energy function. The adaptive HMC algorithm tends to perform well if $D_L$ is contractive during iterations of the Markov chain such as for the German Credit data set as shown in Figure 2. If this is not the case as for the Caravan data in Figure 3, the adapted HMC algorithm can perform worse than MALA or NUTS. More detailed diagnostics for all data sets can be found in Appendix E.




(a) Minimum effective sample size per second (after adaptation). (b) Average acceptance rate (after adaptation). (c) Estimates of the eigenvalues of $D_L$ using power iteration.

Figure 2: Bayesian logistic regression for German credit data set ($d = 25$).




(a) Minimum effective sample size per second (after adaptation). (b) Average acceptance rate (after adaptation). (c) Estimates of the eigenvalues of $D_L$ using power iteration.

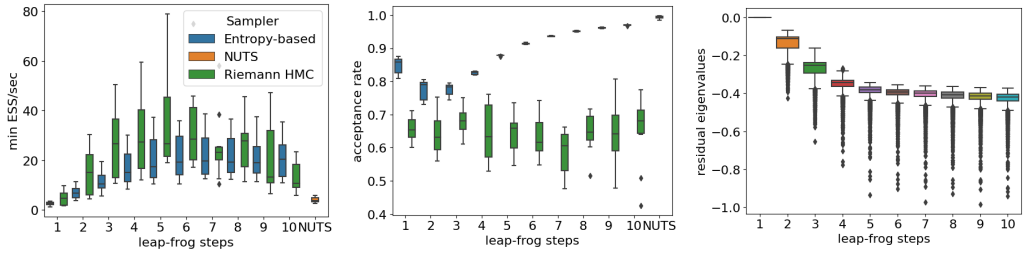
Figure 3: Bayesian logistic regression for caravan data set ($d = 87$).

### 4.3 Log-Gaussian Cox Point Process

We consider inference in a log-Gaussian Cox process model. This is an ideal setting for Riemann-Manifold MALA and HMC [18], since a constant metric tensor is used therein that does not depend on the position, so the complexity is no longer cubic but only quadratic in the dimension $d$ of the target. Consider an area on $[0, 1]^2$ that is discretized into grid locations $(i, j)$, for $i, j = 1, \ldots, n$. The observations $y_{ij}$ are Poisson distributed and conditionally independent given a latent intensity process $\lambda = \{\lambda\}_{ij}$ with means $\lambda_{ij} = m \exp(x_{ij})$ for $m = n^{-2}$ and a latent vector $x$ drawn from a Gaussian process with constant mean function $\mu$ and covariance function $\Sigma_{(i,j),(i',j')} = \sigma_x^2 \exp\{-\sqrt{(i-i')^2 + (j-j')^2}/(n\beta)\}$. The target density is thus proportional to $p(y, x) \propto \prod_{i,j}^{n \times n} \exp\left[y_{ij} x_{ij} - m \exp(x_{ij})\right] \exp\left[-(x - \mu\mathbf{1})^\top \Sigma^{-1} (x - \mu\mathbf{1})/2\right]$. For the Riemann-Manifold based samplers, the preconditioning matrix is $M = \Lambda + \Sigma^{-1}$ where $\Lambda$ is a diagonal matrix with diagonal elements $\{m \exp(\mu + \Sigma_{ii})\}_i$ and we adapt the step size using dual averaging. We generate simulated data for $d \in \{64, 256\}$. We adapt for 2000 steps using a Cholesky factor $C$. Figure 17 in Appendix F illustrates that the gradient-based adaptation can achieve a higher minESS/sec score for $d = 64$ with high acceptance rates for higher leap-frog steps. In dimension $d = 256$, the Riemann-Manifold samplers perform slightly better in terms of minESS/sec, see Figure 4 and Figure 18 for a comparison of the inverse mass matrices.

### 4.4 Stochastic volatility model

We consider a stochastic volatility model [29, 27] that has been used with minor variations for adapting HMC [18, 25, 48]. Assume that the latent log-volatilities follow an autoregressive AR(1)

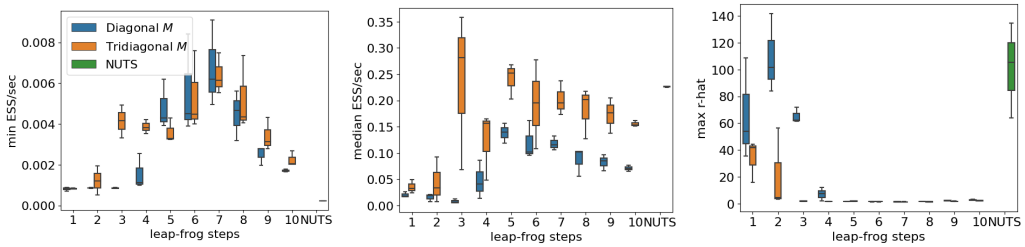



(a) Minimum effective sample size per second (after adaptation). (b) Average acceptance rate (after adaptation). (c) Estimates of the eigenvalues of $D_L$ using power iteration.

Figure 4: Cox process in dimension $d = 256$.

process so that $h_1 \sim \mathcal{N}(0, \sigma^2/(1-\phi^2))$ and for $t \in \{1, \dots, T-1\}$, $h_{t+1} = \phi h_t + \eta_{t+1}$ with $\eta_t \sim \mathcal{N}(0, \sigma^2)$. The observations follow the dynamics $y_t | h_t \sim \mathcal{N}(0, \exp(\mu + h_t))$. The prior distributions for the static parameters are: the persistence of the log-volatility process $(\phi+1)/2 \sim \text{Beta}(20, 1.5)$; the mean log-volatility $\mu \sim \text{Cauchy}(0, 2)$; and the scale of the white-noise process $\sigma \sim \text{Half-Cauchy}(0, 1)$. We reparametrize $\phi$ and $\sigma$ with a sigmoid- and softplus-transformation, respectively. Observe that the precision matrix of the AR(1) process is tridiagonal. Since a Cholesky factor of such a matrix is lower-triangular matrix, we consider the parameterisation $C = B_\theta^{-1}$ for an upper-triangular matrix $B_\theta$. The required operations with such banded matrices have a complexity of $\mathcal{O}(d)$, see for instance [16]. For comparison, we also consider a diagonal matrix $C$. We apply the model to ten years of daily returns of the S&P500 index, giving rise to a target dimension of $d = 2519$. In order to account for the different number of gradient evaluations, we use $3.5 \times 10^4/L$ steps for the adaptation and for evaluating the sampler based on $L \in \{1, \dots, 10\}$ leapfrog steps. We run NUTS for 1000 steps which has a four times higher run-time compared to the other samplers. In addition to using effective sample size to assess convergence, we also report the potential scale reduction factor split-$\hat{R}$ [17, 46] where large values are indicative of poor mixing. We report results over three replications in Figure 5 with more details in Figure 19, Appendix G. First, HMC with moderately large $L$ tends to improve the effective samples per computation time compared to the MALA case, while also having a smaller $\hat{R}$. Second, using a tridiagonal mass matrix improves mixing compared to a diagonal one, particularly for the latent log-volatilities as seen in the median ESS/sec or median $\hat{R}$ values. The largest absolute eigenvalue of $D_L$ tends to be smaller for a tridiagonal mass matrix and the acceptance rates are approaching 100% more slowly for increasing $L$. Third, NUTS seems less efficient as does using a dual-adaptation scheme.

We imagine that similar efficient parametrizations of $M$ or $M^{-1}$ can be used for different generalisations of the above stochastic volatility model, such as including $p$ sub-diagonals for log-volatilities having a higher-order AR($p$) dynamics or multivariate extensions using a suitable block structure. Likewise, this approach might also be useful for inferences in different Gaussian Markov Random Field models with sparse precision matrices.

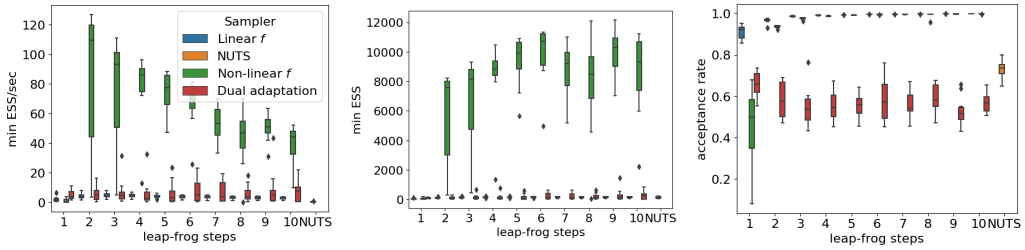



(a) Minimum effective sample size per second (after adaptation). (b) Median effective sample size per second (after adaptation). (c) Maximum $\hat{R}$ of $q \mapsto q_i$ (after adaptation).

Figure 5: Stochastic volatility model ($d = 2519$).

### 4.5 Learning non-linear transformations

To illustrate an extension to sample from highly non-convex targets by learning a non-linear transformation within the suggested framework as explained in greater detail in Appendix C, we consider sampling from a two-dimensional Banana distribution that results from the transformation of $\mathcal{N}(0, \Lambda)$ where $\Lambda$ is a diagonal matrix having entries 100 and 1 via the volume-preserving map $\phi_b(x) = (x_1, x_2 + b(x_1^2 - 100))$, for $b = 0.1$, cf. [20]. We consider a RealNVP-type [13] transformation $f = f_3 \circ f_2 \circ f_1$ where $f_1(x_1, x_2) = (x_1, x_2 \cdot g(s(x_1)) + t(x_1)$, $f_2(x_1, x_2) = (x_1 \cdot g(s(x_2)) + t(x_1), x_2)$ and $f_3(x_1, x_2) = (c_1 x_1, c_2 x_2)$. The functions $s$ and $t$ are neural networks with two hidden layers of size 50. For numerical stability, we found it beneficial to use a modified affine scaling function $g$ as a sigmoid function scaled on a restricted range such as $(0.5, 2)$, as also suggested in [3]. As an alternative, we also consider learning a linear transformation $f(x) = Cx$ for a Cholesky matrix $C$ as well as NUTS and a standard HMC sampler with step size adapted to achieve a target acceptance rate of 0.65. Figure 6 summarizes effective sample sizes where each method uses $4 \times 10^5$ samples before and after the adaptation. Whereas a linear transformation does not improve on standard HMC, applying a non-linear transformation can improve significantly the effective sample size even after taking into account the computational costs.




(a) Minimum effective sample size per second (after adaptation).

(b) Minimum effective sample size (after adaptation).

(c) Average acceptance rate (after adaptation).

Figure 6: Banana-shaped target in dimension $d = 2$.

## 5 Discussion and Outlook

**Convergence.** We have used Adam [30] with a constant step size to adapt the mass matrix, but have stopped the adaptation after some fixed steps so that any convergence is preserved. Different conditions have been established so that infinite adaptive schemes that continue to modify $C$ still converge to the correct invariant distribution, such as diminishing adaptation and containment [42]. An analysis of the convergence properties of the adaptive HMC algorithm is left for future work.

**Limitations.** Our approach to learn a constant mass matrix can struggle for targets where the Hessian varies greatly across the state space, which can yield relatively short integration times with very high acceptance rates. While this effect might be mitigated by considering non-linear transformations, it remains challenging to learn flexible transformations efficiently in high dimensions.

**Variations of HMC.** We have considered a standard HMC setting for a fixed number of leap-frog steps. One could consider a mixture of HMC kernels with different numbers of leap-frog steps and an interesting question would be how to learn the different mass matrices jointly in an efficient way.

Instead of a full velocity refreshment, partial refreshment strategies [26] can sometimes mix better. The suggested adaptation approach can yield very high acceptance rates particularly for increasing leap-frog steps and the learned mass matrix can be used with a partial refreshment. However, it would be interesting to analyse if the adaptation can be adjusted to such persistent velocity updates. It would also be of interest to analyse if similar ideas can be used to adapt different numerical integrators such as those suggested in [4] for target densities relative to a Gaussian measure.

Our focus was on learning a mass matrix so that samples from the Markov chain can be used for estimators that are consistent for increasing iterations. However, unbiased estimators might also be constructed using coupled HMC chains [23] and one might ask if the adapted mass matrix leads to shorter meeting times in such a setting.

# References


[1] Christophe Andrieu, Anthony Lee, and Sam Livingstone. A general perspective on the metropolis-hastings kernel. *arXiv preprint arXiv:2012.14881*, 2020.

[2] Jens Behrmann, Will Grathwohl, Ricky TQ Chen, David Duvenaud, and Joern-Henrik Jacobsen. Invertible residual networks. In *International Conference on Machine Learning*, pages 573–582, 2019.

[3] Jens Behrmann, Paul Vicol, Kuan-Chieh Wang, Roger Grosse, and Jörn-Henrik Jacobsen. Understanding and mitigating exploding inverses in invertible neural networks. In *International Conference on Artificial Intelligence and Statistics*, pages 1792–1800. PMLR, 2021.

[4] Alexandros Beskos, Frank J Pinski, Jesús Marıa Sanz-Serna, and Andrew M Stuart. Hybrid monte carlo on hilbert spaces. *Stochastic Processes and their Applications*, 121(10):2201–2230, 2011.

[5] Alexandros Beskos, Natesh Pillai, Gareth Roberts, Jesus-Maria Sanz-Serna, and Andrew Stuart. Optimal tuning of the hybrid monte carlo algorithm. *Bernoulli*, 19(5A):1501–1534, 2013.

[6] Michael Betancourt. A conceptual introduction to hamiltonian monte carlo. *arXiv preprint arXiv:1701.02434*, 2017.

[7] Nawaf Bou-Rabee and Jesús María Sanz-Serna. Geometric integrators and the hamiltonian monte carlo method. *Acta Numerica*, 27:113–206, 2018.

[8] Tian Qi Chen, Jens Behrmann, David K Duvenaud, and Jörn-Henrik Jacobsen. Residual flows for invertible generative modeling. In *Advances in Neural Information Processing Systems*, pages 9913–9923, 2019.

[9] Yuansi Chen, Raaz Dwivedi, Martin J Wainwright, and Bin Yu. Fast mixing of metropolized hamiltonian monte carlo: Benefits of multi-step gradients. *arXiv preprint arXiv:1905.12247*, 2019.

[10] Zongchen Chen and Santosh S Vempala. Optimal convergence rate of hamiltonian monte carlo for strongly logconcave distributions. *arXiv preprint arXiv:1905.02313*, 2019.

[11] Nicolas Chopin, James Ridgway, et al. Leave pima indians alone: binary regression as a benchmark for bayesian computation. *Statistical Science*, 32(1):64–87, 2017.

[12] Rob Cornish, Anthony L Caterini, George Deligiannidis, and Arnaud Doucet. Relaxing bijectivity constraints with continuously indexed normalising flows. *arXiv preprint arXiv:1909.13833*, 2019.

[13] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016.

[14] Simon Duane, Anthony D Kennedy, Brian J Pendleton, and Duncan Roweth. Hybrid monte carlo. *Physics letters B*, 195(2):216–222, 1987.

[15] Alain Durmus, Eric Moulines, and Eero Saksman. On the convergence of hamiltonian monte carlo. *arXiv preprint arXiv:1705.00166*, 2017.

[16] Nicolas Durrande, Vincent Adam, Lucas Bordeaux, Stefanos Eleftheriadis, and James Hensman. Banded matrix operators for gaussian markov models in the automatic differentiation era. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 2780–2789. PMLR, 2019.

[17] Andrew Gelman, John B Carlin, Hal S Stern, David B Dunson, Aki Vehtari, and Donald B Rubin. *Bayesian data analysis*. CRC press, 2013.

[18] Mark Girolami and Ben Calderhead. Riemann manifold langevin and hamiltonian monte carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(2): 123–214, 2011.

[19] Gene H Golub and Charles F Van Loan. *Matrix computations*, volume 3. JHU Press, 2012.

[20] Heikki Haario, Eero Saksman, and Johanna Tamminen. Adaptive proposal distribution for random walk metropolis algorithm. *Computational Statistics*, 14(3):375–395, 1999.

[21] Ernst Hairer, Christian Lubich, and Gerhard Wanner. Geometric numerical integration illustrated by the störmer–verlet method. *Acta numerica*, 12:399–450, 2003.

[22] Insu Han, Haim Avron, and Jinwoo Shin. Stochastic chebyshev gradient descent for spectral optimization. In *Advances in Neural Information Processing Systems*, pages 7386–7396, 2018.

[23] Jeremy Heng and Pierre E Jacob. Unbiased hamiltonian monte carlo with couplings. *Biometrika*, 106(2):287–302, 2019.

[24] Matthew Hoffman, Pavel Sountsov, Joshua V Dillon, Ian Langmore, Dustin Tran, and Srinivas Vasudevan. Neutra-lizing bad geometry in hamiltonian monte carlo using neural transport. *arXiv preprint arXiv:1903.03704*, 2019.

[25] Matthew D Hoffman and Andrew Gelman. The no-u-turn sampler: adaptively setting path lengths in hamiltonian monte carlo. *Journal of Machine Learning Research*, 15(1):1593–1623, 2014.

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

[27] Eric Jacquier, Nicholas G Polson, and Peter E Rossi. Bayesian analysis of stochastic volatility models. *Journal of Business & Economic Statistics*, 20(1):69–87, 2002.

[28] Leif T Johnson and Charles J Geyer. Variable transformation to obtain geometric ergodicity in the random-walk metropolis algorithm. *The Annals of Statistics*, 40(6):3050–3076, 2012.

[29] Sangjoon Kim, Neil Shephard, and Siddhartha Chib. Stochastic volatility: likelihood inference and comparison with arch models. *The review of economic studies*, 65(3):361–393, 1998.

[30] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[31] Ian Langmore, Michael Dikovsky, Scott Geraedts, Peter Norgaard, and Rob Von Behren. A condition number for hamiltonian monte carlo. *arXiv preprint arXiv:1905.09813*, 2019.

[32] Junpeng Lao, Christopher Suter, Ian Langmore, Cyril Chimisov, Ashish Saxena, Pavel Sountsov, Dave Moore, Rif A Saurous, Matthew D Hoffman, and Joshua V Dillon. tfp. mcmc: Modern markov chain monte carlo tools built for modern hardware. *arXiv preprint arXiv:2002.01184*, 2020.

[33] Daniel Levy, Matt D Hoffman, and Jascha Sohl-Dickstein. Generalizing hamiltonian monte carlo with neural networks. In *International Conference on Learning Representations*, 2018.

[34] Zengyi Li, Yubei Chen, and Friedrich T Sommer. A neural network mcmc sampler that maximizes proposal entropy. *arXiv preprint arXiv:2010.03587*, 2020.

[35] Samuel Livingstone, Michael Betancourt, Simon Byrne, and Mark Girolami. On the geometric ergodicity of hamiltonian monte carlo. *Bernoulli*, 25(4A):3109–3138, 2019.

[36] Samuel Livingstone, Michael F Faulkner, and Gareth O Roberts. Kinetic energy choice in hamiltonian/hybrid monte carlo. *Biometrika*, 106(2):303–319, 2019.

[37] Anne-Marie Lyne, Mark Girolami, Yves Atchadé, Heiko Strathmann, Daniel Simpson, et al. On russian roulette estimates for bayesian inference with doubly-intractable likelihoods. *Statistical science*, 30(4):443–467, 2015.

[38] Oren Mangoubi and Aaron Smith. Rapid mixing of hamiltonian monte carlo on strongly log-concave distributions. *arXiv preprint arXiv:1708.07114*, 2017.

[39] Radford M Neal. Mcmc using hamiltonian dynamics. *Handbook of markov chain monte carlo*, 2(11):2, 2011.

[40] Joonha Park and Yves Atchadé. Markov chain monte carlo algorithms with sequential proposals. *Statistics and Computing*, 30(5):1325–1345, 2020.

[41] Cristian Pasarica and Andrew Gelman. Adaptively scaling the metropolis algorithm using expected squared jumped distance. *Statistica Sinica*, pages 343–364, 2010.

[42] Gareth O Roberts and Jeffrey S Rosenthal. Coupling and ergodicity of adaptive markov chain monte carlo algorithms. *Journal of applied probability*, 44(2):458–475, 2007.

[43] Jascha Sohl-Dickstein, Mayur Mudigonda, and Michael DeWeese. Hamiltonian monte carlo without detailed balance. In *International Conference on Machine Learning*, pages 719–726, 2014.

[44] Pavel Sountsov, Alexey Radul, and contributors. Inference gym, 2020. URL https://pypi.org/project/inference_gym.

[45] Michalis Titsias and Petros Dellaportas. Gradient-based adaptive markov chain monte carlo. In *Advances in Neural Information Processing Systems*, pages 15704–15713, 2019.

[46] Aki Vehtari, Andrew Gelman, Daniel Simpson, Bob Carpenter, and Paul-Christian Bürkner. Rank-normalization, folding, and localization: An improved r for assessing convergence of mcmc. *Bayesian analysis*, 1(1):1–28, 2021.

[47] Ziyu Wang, Shakir Mohamed, and Nando Freitas. Adaptive hamiltonian and riemann manifold monte carlo. In *International conference on machine learning*, pages 1462–1470, 2013.

[48] Changye Wu, Julien Stoehr, and Christian P Robert. Faster hamiltonian monte carlo by learning leapfrog scale. *arXiv preprint arXiv:1810.04449*, 2018.

[49] Tengchao Yu, Hongqiao Wang, and Jinglai Li. Maximum conditional entropy hamiltonian monte carlo sampler. *arXiv preprint arXiv:1910.05275*, 2019.

# Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [No] Although probabilistic models that require efficient inference methods such as HMC can be utilized in different decision making processes, we do not see a direct path to applications with a negative societal impact.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] Code to reproduce the main results will be added to the supplementary material.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] We have specified the major hyperparameters in the paper. Minor choices for the hyperparameters can be found in the code.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No] We generally used 10 replications for all experiments to create the box-plots for assessing the mixing properties, with the exception of the stochastic volatility model that uses only three replications.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [No] We have included the CPU time for running the adapted Markov chains in our paper. Run times for the adaptation are of similar order. We run our experiments on an internal cluster using mostly single-core CPUs (Intel(R) Xeon(R) Gold 6240 CPU @ 2.60GHz) with the exception of the stochastic volatility experiments that use four cores.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] See [32, 44].
   (b) Did you mention the license of the assets? [No]
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [No]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [No]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]