Entropy-based adaptive Hamiltonian Monte Carlo

Marcel Hirt
Department of Statistical Science
University College London, UK
marcel.hirt.16@ucl.ac.uk

Michalis K. Titsias
DeepMind
London, UK
mtitsias@google.com

Petros Dellaportas
Department of Statistical Science
University College London, UK
Department of Statistics,
Univ. of Econ. and Business, Athens, Greece
and The Alan Turing Institute, UK

Abstract

Hamiltonian Monte Carlo (HMC) is a popular Markov Chain Monte Carlo (MCMC) algorithm to sample from an unnormalized probability distribution. A leapfrog integrator is commonly used to implement HMC in practice, but its performance can be sensitive to the choice of mass matrix used therein. We develop a gradient-based algorithm that allows for the adaptation of the mass matrix by encouraging the leapfrog integrator to have high acceptance rates while also exploring all dimensions jointly. In contrast to previous work that adapt the hyper-parameters 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. 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 π 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 [20, 46, 9] 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, φ_t maps each (q,p) to the solution of (1) that takes value (q,p) at time $t=0$. The exact HMC flow φ preserves volume and conserves the total energy *i.e.* $H \circ \varphi_t = H$. Consequently, the Boltzmann-Gibbs distribution μ 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 μ 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 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 [28, 10]. 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 $0 \leqslant \ell \leqslant L-1$,

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

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 φ, 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 μ-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 $a(q_0,p_0) = \min\{1, \exp\left[-\left(H(q_L,p_L) - H(q_0,p_0)\right)\right]\}$, 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.

We want to explore here further the generalised speed measure introduced in [54] 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 [41] 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 [19]. [60] 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.

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 $d \to \infty$ [8] for a given integration time. HMC hyperparameters have been tuned using some form of *expected squared jumping distance* (ESJD) [49], using for instance Bayesian optimization [56] or a gradient-based approach [40]. A popular approach suggested in [32] 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 [4]; see also [57] for a variation and [48] for a version using sequential proposals. Modern probabilistic programming languages such as Stan [12], PyMC3 [51], Turing [23, 58] or TFP [39] furthermore allow for an adaptation of a diagonal or dense mass-matrix within NUTS based on the sample covariance matrix. The Riemann manifold HMC algorithm from [25] 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 [43] by analysing the behaviour of $x \mapsto \nabla K(\nabla U(x))$. Different pre-conditioning approaches have been compared for Gaussian targets in [38]. A popular route has also been to first transform the target using tools from variational inference as in [31] 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, [45, 15] 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 κ, assuming the integration time is set to $T = \frac{1}{2\sqrt{m_2}}$. [14] 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) an overlap bound, that is 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'. While such bounds for controlling the mixing time might share some similarity with the generalised speed measure, they do not lend themselves easily to a gradient-based adaptation.

3 Entropy-based adaptation scheme

We derive a novel method to approximate the entropy of the proposed position after L leapfrog steps. Our approximation is based on the assumption that the Hessian of the target is locally constant around the mid-point of the HMC trajectory. This allows for a fast stochastic trace estimator of the marginal proposal entropy. We then develop a penalised loss function that can be minimized using stochastic gradient descent while sampling from the Markov chain in order to optimize a generalised speed measure.

3.1 Marginal proposal entropy

Suppose that $CC^\top = M^{-1}$, where C is defined by some parameters θ 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 reparameterize 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

$$q_L = \mathcal{T}_L(v) = q_0 - \frac{Lh^2}{2} M^{-1} \nabla U(q_0) + LhCv - h^2 M^{-1} \Xi_L(v), \tag{3}$$

and

$$p_L = \mathcal{W}_L(v) = C^{-\top} v - \frac{h}{2}\left[\nabla U(q_0) + \nabla U \circ \mathcal{T}_L(v)\right] - h \sum_{i=1}^{L-1} \nabla U \circ \mathcal{T}_i(v) \tag{4}$$

where

$$\Xi_L(v) = \sum_{i=1}^{L-1} (L-i) \nabla U \circ \mathcal{T}_i(v), \tag{5}$$

see also [42, 21, 14] 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 [21], the transformation $\mathcal{T}_L \colon \mathbb{R}^d \to \mathbb{R}^d$ is a C^1-diffeomorphism. With ν 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 ν via the map $\mathcal{T}_L$ so that[1]

$$\log r_L(\mathcal{T}_L(v)) = \log \nu(v) - \log|\det \mathsf{D}\mathcal{T}_L(v)|. \tag{6}$$

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

$$\mathcal{S}_L \colon \mathbb{R}^d \to \mathbb{R}^d,\ v \mapsto \frac{1}{Lh} C^{-1} \mathcal{T}_L(v) - v. \tag{7}$$

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

$$\log|\det \mathsf{D}\mathcal{T}_L(v)| = d\log(Lh) + \log|\det C| + \log|\det(\mathrm{I} + \mathsf{D}\mathcal{S}_L(v))|. \tag{8}$$

Combining (6) and (8) yields the log-probability of the HMC proposal

$$\log r_L(\mathcal{T}_L(v)) = \log \nu(v) - d\log(Lh) - \log|\det C| - \log|\det(\mathrm{I} + \mathsf{D}\mathcal{S}_L(v))|. \tag{9}$$

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

Comparing the equations (3) and (7), 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 θ 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 $\ell \in \{2, \ldots, L\}$, $v \in \mathbb{R}^d$,*

$$\mathsf{D}\mathcal{S}_\ell(v) = -h^2 \sum_{i=1}^{\ell-1} (\ell - i) \frac{i}{\ell} C^\top \nabla^2 U \left(\mathcal{T}_i(v)\right) C \left(\mathrm{I} + \mathsf{D}\mathcal{S}_i(v)\right). \tag{10}$$

In particular, $\mathsf{D}\mathcal{S}_\ell(v)$ is a symmetric matrix. Suppose further that $L^2 h^2 < \sup_{q \in \mathbb{R}^d} \frac{1}{4\|C^\top \nabla^2 U(q) C\|_2}$. Then for any $\ell \in \{1, \ldots, L\}$ and $v \in \mathbb{R}^d$, we have $\|\mathsf{D}\mathcal{S}_\ell(v)\|_2 < \frac{1}{8}$.

Notice that the recursive formula (10) requires computing $\frac{1}{2}L(L-1)$ terms, each involving the Hessian, in order to compute the Jacobian after L leapfrog steps. 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 (10), for any $q \in \mathbb{R}^d$, $v \in \mathbb{R}^d$,

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

where

$$D_L = -h^2 C^\top \Sigma^{-1} C \left(\sum_{i=1}^{L-1} (L - i) \frac{i}{L} \right) = -h^2 \frac{L^2 - 1}{6} C^\top \Sigma^{-1} C \tag{11}$$

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 $\left\|C^\top \Sigma^{-1} C\right\|_2 \leqslant \frac{1}{4h^2 L^2}$, 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. Put differently, for Gaussian targets, $\mathsf{D}\mathcal{S}_L$ can be decomposed into a component D_L that contains all terms that are linear in $h^2 C^\top \Sigma^{-1} C$ and that does not require a recursion; plus a component R_L that contains terms that are higher than linear in $h^2 C^\top \Sigma^{-1} C$ and that needs to be solved recursively. Our suggestion is to ignore this second term. Notice that $R_2 = 0$ and an extension can be to 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.

For an arbitrary potential energy U, equation (10) shows that evaluating $\mathsf{D}\mathcal{S}_L$ leads to a non-linear function of the Hessians evaluated along the different points of the leapfrog-trajectory. We suggest to replace it with a first order term with one Hessian evaluation which is however scaled accordingly. Concretely, we maximize

$$\mathcal{L}(\theta) = \log|\det(\mathrm{I} + D_L)| \quad \text{with} \quad D_L = -h^2 \frac{L^2 - 1}{6} C^\top \nabla^2 U(q_{\lfloor L/2 \rfloor}) C \tag{12}$$

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 Σ^{-1} in (11) given by the Hessian of U at the mid-point $q_{\lfloor L/2 \rfloor}$ of the trajectory. We want to optimize $\mathcal{L}(\theta)$ given in (12) 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 [44], see also [29, 5, 13] 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,

$$\mathcal{L}(\theta) = \log \det(\mathrm{I} + D_L) = \mathbb{E}_{N, \varepsilon} \left[\sum_{k=1}^{N} \frac{(-1)^{k+1}}{k p_k} \varepsilon^\top (D_L)^k \varepsilon \right], \tag{13}$$

where ε 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 [44, 17]. 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 [26]. 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 [54] to adapt the HMC proposal, we aim to solve

$$\arg\min_{\theta} \int \int \pi(q_0)\nu(v) \Big[-\log a\left((q_0, v), (\mathcal{T}_L(v), \mathcal{W}_L(v))\right) + \beta \log r_L(\mathcal{T}_L(v)) \Big] \mathrm{d}v \mathrm{d}q_0, \tag{14}$$

where $\mathcal{T}_L$, $\mathcal{W}_L$, r_L as well as the acceptance rate a depend on q_0 and the parameters θ we want to adapt. Also, the hyper-parameter $\beta > 0$ can be adapted online by increasing β if the acceptance rate is above a target acceptance rate $\alpha_\star$ and decreasing β otherwise. We choose $\alpha_\star = 0.67$, which is optimal for increasing d under independence assumptions [8]. 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 [54] tends to improve the adaptation by stopping gradients through ∇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 (14) 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 β and γ. The adaptation scheme in Algorithm 1 requires to choose learning rates ρ_θ, ρ_β, ρ_γ and can be viewed within a stochastic approximation framework of controlled Markov chains, see for instance [2, 1, 3]. Different conditions have been established so that infinite adaptive schemes still converge to the correct invariant distribution, such as diminishing adaptation and containment [50]. We have used Adam [37] 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 and we leave an investigation of convergence properties of an infinite adaptive scheme for future work.

4 Numerical experiments

This section illustrates the mixing performance of the entropy-based sampler for a variety of target densities. First, we consider Gaussian targets either in high dimensions or with a high condition number. Our results confirm (i) that HMC scales better than MALA for high-dimensional Gaussian targets and (ii) that the adaptation scheme learns a mass matrix that is adjusted to the geometry of the target. This is in contrast to adaptation schemes trying to optimize the ESJD [49] or variants thereof [40] that can lead to good mixing in a few components only. Next, we apply the novel adaptation scheme to Bayesian logistic regression models and find that it often outperforms NUTS, except in a few data sets where some components might mix less efficiently. We also compare the entropy-based adaptation with Riemann-Manifold based samplers for a Log-Gaussian Cox point process models. We find that both schemes mix similarly, which indicates that the gradient-based adaptation scheme can learn a suitable mass matrix without having access to the expected Fisher information matrix. Then, we consider a high-dimensional stochastic volatility model where the entropy-based scheme performs favourably compared to alternatives and illustrate that efficient sparsity assumptions can be accommodated when learning the mass matrix. Finally, we show in a toy example how the suggested

Algorithm 1 Sample the next state q' and adapt β, γ and θ.

```
1: Sample velocity $v \sim \mathcal{N}(0, \mathrm{I})$ and set $p = C^{-\top} v$.
2: Apply integrator LF to obtain $(q_\ell, p_\ell, \nabla U(q_\ell))_{0 \leqslant \ell \leqslant L} = \mathtt{LF}(q, p)$.
3: Stop gradients $\nabla U(q_\ell) = \mathtt{stop\_grad}(\nabla U(q_\ell))$ for $0 \leqslant \ell \leqslant L$.
4: Compute $\Xi_L(v)$ using (5).
5: Compute $\Delta(q_0, v)$ using (16) and set $a = \min\{1, \mathrm{e}^{-\Delta(q_0,v)}\}$.
6: Compute $\bar{\eta}_N, y = \textsc{Rademacher}()$.
7: Set $\mathcal{L}(\theta) = \mathtt{stop\_grad}(y)^\top D_L \varepsilon$.
8: Set $b_N = \mathtt{stop\_grad}\left(\frac{\bar{\eta}_N}{\|\bar{\eta}_N\|_2^2}\right)$ and $\mu_N = b_N^\top D_L b_N$.
9: $\mathcal{E}(\theta) = -\min\{0, -\Delta(q_0, v)\} - \beta\left(d \log h + \log|\det C| + \mathcal{L}(\theta) - \gamma h(|\mu_N|)\right)$.
10: Adapt $\theta \leftarrow \theta - \rho_\theta \nabla_\theta \mathcal{E}(\theta)$.
11: Adapt $\beta \leftarrow \Pi_\beta\left[\beta(1 + \rho_\beta(a - \alpha_\star)\right]$. #$\Pi_\beta$ projects onto a compact set; default value $[10^{-2}, 10^2]$.
12: Adapt $\gamma \leftarrow \Pi_\gamma\left[\gamma + \rho_\gamma h(|\mu_N|)\right]$. #$\Pi_\gamma$ projects onto a compact set; default value $[10^3, 10^5]$.
13: Sample $u \sim \mathcal{U}(0, 1)$ and set $q' = \mathbf{1}_{\{u \leqslant a\}}\, q_L + \mathbf{1}_{\{u > a\}}\, q$.

14: function $D_L(w)$:
15:     #$D_L(w) = D_L w$ computes Hessian-vector products efficiently
16:     $z = \mathtt{vjp}(\nabla U, \mathtt{stop\_grad}(q_{\lfloor L/2 \rfloor}), C w)^\top$
17:     return $-h^2 \frac{L^2-1}{6}\, C^\top z$
18: end function

19: function Rademacher:
20:     Sample Rademacher random variable $\varepsilon$ and truncation level $N$.
21:     Initialise $y \leftarrow 0$ and $\bar{\eta}_0 = \varepsilon$.
22:     for $k = 1 \ldots N$ do
23:         #Apply a spectral normalisation for stability if $D_L$ is not a contraction; $\delta' \in (0, 1)$.
24:         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 $y \leftarrow y + \frac{(-1)^k}{p_k} \bar{\eta}_k$.
25:     end for
26:     return $\bar{\eta}_N, y$
27: end function
```

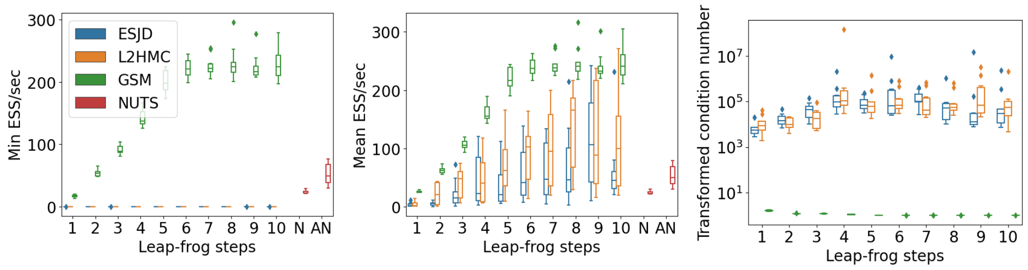
approach might be modified to sample from highly non-convex potentials. Our implementation[2] builds up on tensorflow probability [39] with some target densities taken from [53]. We used 10 parallel chains throughout our experiments to adapt the mass matrix.

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 Σ. 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 [52]. 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 4×10^4 steps in case (i) and for 10^5 steps in case (ii). We compared it with a NUTS implementation in tensorflow probability (TFP) [39] with a default maximum tree depth of 10 and step sizes adapted using dual averaging [32, 47] that we denote by N in the figures below. Additionally, we consider a further adaptation of NUTS by adapting a diagonal mass matrix using an online variance estimate of the accepted samples as implemented in TFP and denoted AN subsequently. We also consider two objectives as a replacement of the generalised speed measure (GSM): (a) the ESJD and (b) a weighted combination of the ESJD and its inverse as suggested in Levy et al. [40], without any burn-in component, which we denote L2HMC, see Appendix D for a precise definition. We compute the minimum and mean effective sample size (minESS and meanESS) of all functions $q \mapsto q_i$ over $i \in \{1, \ldots, d\}$ as shown in Figure 1a-1b for $d = 10^3$ in case (i) with leapfrog steps ranging from $L = 1$ to 10. It can be observed that HMC adapted with the GSM objective performs

[2] https://github.com/marcelah/entropy_adaptive_hmc

well in terms of minESS/sec for $L > 1$, whereas the ESJD or L2HMC objectives yield poor mixing as measured in terms of the minESS/sec. The meanESS/sec statistics are more similar for the different objectives. These observations provide some empirical evidence that the ESJD can be high even when some components mix poorly, which has been a major motivation for the GSM objective in [54]. The mass matrix learned using the GSM adapts to the target covariance as can be seen from the the condition numbers of $C^\top \Sigma^{-1} C$ in Figure 1c becoming relatively close to 1. The GSM objective also yields acceptance rates approaching 1 for increasing leap-frog steps and multiplication with D_L becomes a contraction as shown in Appendix F.1, Figure 7. Results for $d = 10^4$ can be found in Figure 8 in Appendix F.1 which indicate that as the dimension increases, using more leap-frog steps becomes more advantageous. For the case (ii) of a very ill-conditioned target, results in Table 1 show that the GSM objective leads to better minESS/sec values, while further statistics shown in Figure 9 illustrate that the GSM also yields to higher minESS/sec values compared to NUTS with an adapted mass matrix. We want to emphasize that for fixed L, high acceptance rates for HMC need not be disadvantageous. This is illustrated in Figure 11 in Appendix F.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.



(a) (b) (c)

Figure 1: Minimum (1a) and mean (1b) effective sample size of $q \mapsto q_i$ per second after adaptation for an anisotropic Gaussian target ($d = 1000$). The condition number of the transformed Hessian $C^\top \Sigma^{-1} C$ are shown in (1c).

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 [54], 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 10 in Appendix F.3 and summarized in Table 2.

Table 1: MinESS/sec for gradient-based adaptation schemes targeting an ill-conditioned Gaussian density ($d = 100$).

Steps	GSM	ESJD	L2HMC
1	122.3 (15.5)	0.1 (0.01)	0.1 (0.01)
5	753.8 (22.2)	0.1 (0.02)	0.1 (0.02)
10	570.0 (37.4)	0.6 (395.2)	0.1 (0.05)

Table 2: MinESS/sec for gradient-based adaptation schemes targeting a correlated Gaussian density ($d = 51$).

Steps	GSM	ESJD	L2HMC
1	63.8 (3.9)	0.8 (1.6)	0.3 (0.1)
5	390.0 (5.0)	2.0 (5.4)	2.7 (2.3)
10	282.7 (7.8)	0.9 (3.7)	0.4 (0.9)

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 Σ_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. [16]. The state dimension ranges from $d = 3$ to $d = 87$. We choose $\Sigma_0 = \mathrm{I}$ and parameterize 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 mean ESS/sec across dimensions. 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, where the eigenvalues of D_L are estimated using a power iteration. 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 G.

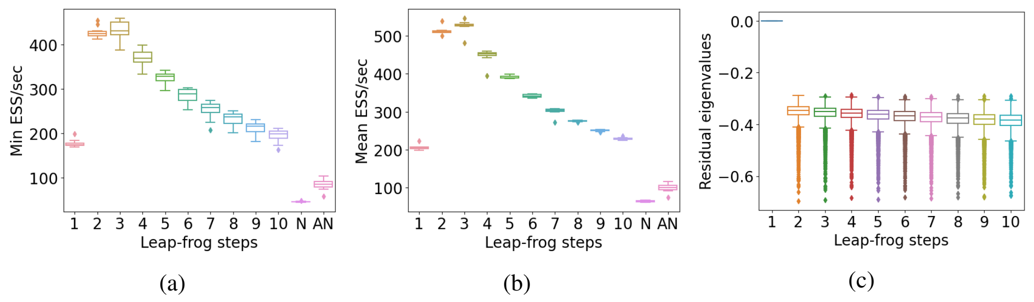


Figure 2: Minimum (2a) and mean (2b) effective sample size for a Bayesian logistic regression model for German credit data set ($d = 25$) after adaptation. Estimates of the eigenvalues of D_L 2c.

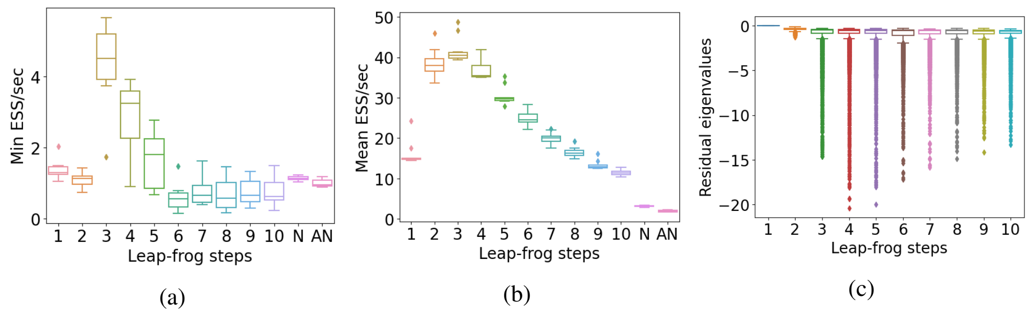


Figure 3: Minimum (3a) and mean (3b) effective sample size for a Bayesian logistic regression model for caravan data set ($d = 87$) after adaptation. Estimates of the eigenvalues of D_L 3c.

4.3 Log-Gaussian Cox Point Process

Inference in a log-Gaussian Cox process model is an ideal setting for Riemann-Manifold (RM) MALA and HMC [25], as a constant metric tensor is used therein that does not depend on the position, making the complexity no longer cubic but only quadratic in the dimension d of the target. Consider an area on $[0,1]^2$ discretized into grid locations (i,j), for $i,j = 1,\dots,n$. The observations y_{ij} are Poisson distributed and conditionally independent given a latent intensity process $\{\lambda\}_{ij}$ 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 μ 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 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 RM based samplers, the preconditioning matrix is $M = \Lambda + \Sigma^{-1}$ where Λ is a diagonal matrix with diagonal elements $\{m\exp(\mu + \Sigma_{ii})\}_i$ and step sizes adapted using dual averaging. We generate simulated data for $d \in \{64, 256\}$ and adapt for 2000 steps using a Cholesky factor C. Figure 18 in Appendix H illustrates that the entropy-based adaptation can achieve a higher minESS/sec score for $d = 64$ with high acceptance rates for increasing leap-frog steps. The RM samplers perform slightly better in terms of minESS/sec for $d = 256$, see Figure 4 and Figure 19 for a comparison of the inverse mass matrices.

4.4 Stochastic volatility model

We consider a stochastic volatility model [36, 34] that has been used with minor variations for adapting HMC [25, 32, 57]. Assume that the latent log-volatilities follow an autoregressive AR(1)

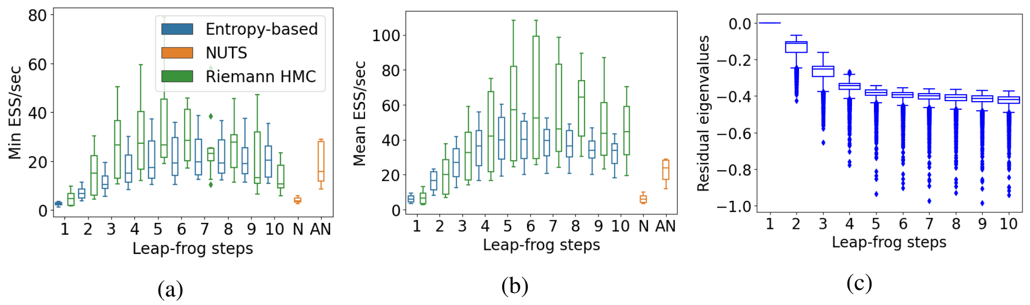


Figure 4: Minimum (4a) and mean (4b) effective sample size for a Cox process in dimension $d = 256$ after adaptation. Estimates of the eigenvalues of D_L using power iteration in (4c).

process so that $h_1 \sim \mathcal{N}(0, \sigma^2/(1-\phi^2))$ and for $t \in \{1, \ldots, T-1\}$, $h_{t+1} = \phi h_t + \eta_{t+1}$ 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 ϕ and σ 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 tridiagonal, we consider the parameterization $C = B_\theta^{-1}$ for an upper-triangular and tridiagonal matrix B_θ. The required operations with such banded matrices have a complexity of $\mathcal{O}(d)$, see for instance [22]. 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 $L \in \{1, \ldots, 10\}$ 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-$\hat{R}$ [24, 55] where large values are indicative of poor mixing. We report results over three replications in Figure 5 with more details in Figure 20, Appendix I. 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 parameterizations 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.

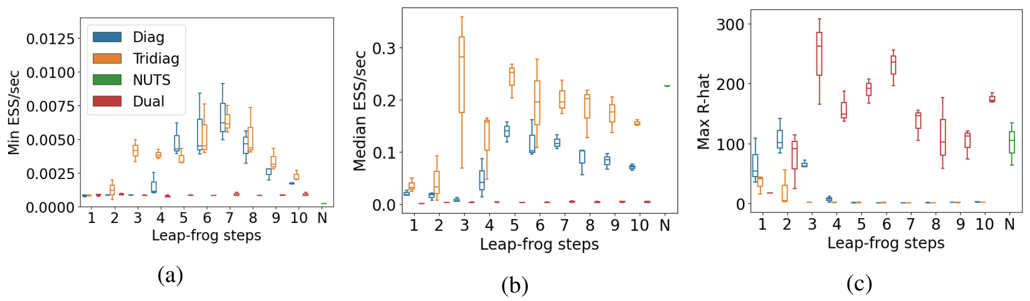
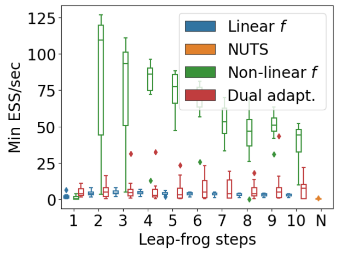


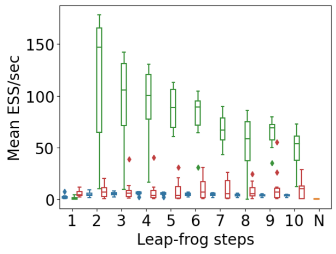
Figure 5: Minimum (5a) and median (5b) effective sample size per second and maximum $\hat{R}$ of $q \mapsto q_i$ for a stochastic volatility model ($d = 2519$) after adaptation.

4.5 Learning non-linear transformations

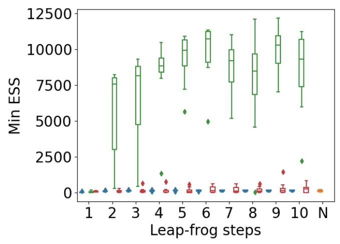
To illustrate an extension to sample from highly non-convex targets by learning a non-linear transformation within the suggested framework as explained in greater detail in Appendix C, we consider sampling from a two-dimensional Banana distribution that results from the transformation of $\mathcal{N}(0, \Lambda)$ where Λ 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. [27]. We consider a RealNVP-type [19] 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 [6]. 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 the ESS where each method uses 4×10^5 samples before and after the adaptation. Whereas a linear transformation does not improve on standard HMC, non-linear transformations can improve the mixing efficiency.



(a)



(b)



(c)

Figure 6: Minimum (6a) and mean (6b) effective sample size per second as well as minimum effective sample size (6c) for a Banana-shaped target in dimension $d = 2$ after adaptation.

5 Discussion and Outlook

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 the entropy objective. Recent work [18, 11] have suggested to add the cross-entropy term $\int \pi(q) \int r(q'|q) \log \pi(q') \mathrm{d}q' \mathrm{d}q$ to the entropy objective for optimizing the parameters of a Metropolis-Hastings kernel with proposal density $r(q'|q)$. Algorithm 1 can be adjusted to such variations, possibly by stopping gradients through ∇U as for optimizing the energy error term.

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 [33] 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 [7] for target densities relative to a Gaussian measure or for multinomial HMC with an additional intra-trajectory sampling step [9, 59].

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 [30] and one might ask if the adapted mass matrix leads to shorter meeting times in such a setting.

Acknowledgements

The authors acknowledge the use of the UCL Myriad High Performance Computing Facility (Myriad@UCL), and associated support services, in the completion of this work.

References

[1] Christophe Andrieu and Éric Moulines. On the ergodicity properties of some adaptive MCMC algorithms. *The Annals of Applied Probability*, 16(3):1462–1505, 2006.

[2] Christophe Andrieu and Johannes Thoms. A tutorial on adaptive MCMC. *Statistics and computing*, 18(4):343–373, 2008.

[3] Christophe Andrieu, Vladislav B Tadic, and Matti Vihola. On the stability of some controlled Markov chains and its applications to stochastic approximation with Markovian dynamic. *The Annals of Applied Probability*, 25(1):1–45, 2015.

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

[5] 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.

[6] 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.

[7] Alexandros Beskos, Frank J Pinski, Jesús Maria Sanz-Serna, and Andrew M Stuart. Hybrid Monte Carlo on Hilbert spaces. *Stochastic Processes and their Applications*, 121(10):2201–2230, 2011.

[8] 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.

[9] Michael Betancourt. A conceptual introduction to Hamiltonian Monte Carlo. *arXiv preprint arXiv:1701.02434*, 2017.

[10] Nawaf Bou-Rabee and Jesús Maria Sanz-Serna. Geometric integrators and the Hamiltonian Monte Carlo method. *Acta Numerica*, 27:113–206, 2018.

[11] Chris Cannella and Vahid Tarokh. Semi-Empirical Objective Functions for MCMC Proposal Optimization. *arXiv preprint arXiv:2106.02104*, 2021.

[12] Bob Carpenter, Andrew Gelman, Matthew D Hoffman, Daniel Lee, Ben Goodrich, Michael Betancourt, Marcus Brubaker, Jiqiang Guo, Peter Li, and Allen Riddell. Stan: A probabilistic programming language. *Journal of statistical software*, 76(1):1–32, 2017.

[13] 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.

[14] 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.

[15] Zongchen Chen and Santosh S Vempala. Optimal convergence rate of Hamiltonian Monte Carlo for strongly logconcave distributions. *arXiv preprint arXiv:1905.02313*, 2019.

[16] 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.

[17] Rob Cornish, Anthony L Caterini, George Deligiannidis, and Arnaud Doucet. Relaxing Bijectivity Constraints with Continuously Indexed Normalising Flows. *arXiv preprint arXiv:1909.13833*, 2019.

[18] Ameer Dharamshi, Vivian Ngo, and Jeffrey S Rosenthal. Sampling by Divergence Minimization. *arXiv preprint arXiv:2105.00520*, 2021.

[19] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using Real NVP. *arXiv preprint arXiv:1605.08803*, 2016.

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

[21] Alain Durmus, Eric Moulines, and Eero Saksman. On the convergence of Hamiltonian Monte Carlo. *arXiv preprint arXiv:1705.00166*, 2017.

[22] 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.

[23] Hong Ge, Kai Xu, and Zoubin Ghahramani. Turing: A language for flexible probabilistic inference. In *International conference on artificial intelligence and statistics*, pages 1682–1690. PMLR, 2018.

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

[25] 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.

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

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

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

[29] 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.

[30] Jeremy Heng and Pierre E Jacob. Unbiased Hamiltonian Monte Carlo with couplings. *Biometrika*, 106(2):287–302, 2019.

[31] 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.

[32] 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.

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

[34] 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.

[35] 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.

[36] 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.

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

[38] 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.

[39] 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.

[40] Daniel Levy, Matt D Hoffman, and Jascha Sohl-Dickstein. Generalizing Hamiltonian Monte Carlo with neural networks. In *International Conference on Learning Representations*, 2018.

[41] Zengyi Li, Yubei Chen, and Friedrich T Sommer. A Neural Network MCMC sampler that maximizes Proposal Entropy. *arXiv preprint arXiv:2010.03587*, 2020.

[42] Samuel Livingstone, Michael Betancourt, Simon Byrne, and Mark Girolami. On the geometric ergodicity of Hamiltonian Monte Carlo. *Bernoulli*, 25(4A):3109–3138, 2019.

[43] Samuel Livingstone, Michael F Faulkner, and Gareth O Roberts. Kinetic energy choice in Hamiltonian/hybrid Monte Carlo. *Biometrika*, 106(2):303–319, 2019.

[44] 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.

[45] Oren Mangoubi and Aaron Smith. Rapid mixing of Hamiltonian Monte Carlo on strongly log-concave distributions. *arXiv preprint arXiv:1708.07114*, 2017.

[46] Radford M Neal. MCMC using Hamiltonian dynamics. *Handbook of markov chain monte carlo*, 2(11):2, 2011.

[47] Yurii Nesterov. Primal-dual subgradient methods for convex problems. *Mathematical programming*, 120(1):221–259, 2009.

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

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

[50] 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.

[51] John Salvatier, Thomas V Wiecki, and Christopher Fonnesbeck. Probabilistic programming in Python using PyMC3. *PeerJ Computer Science*, 2:e55, 2016.

[52] Jascha Sohl-Dickstein, Mayur Mudigonda, and Michael DeWeese. Hamiltonian Monte Carlo Without Detailed Balance. In *International Conference on Machine Learning*, pages 719–726, 2014.

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

[54] Michalis Titsias and Petros Dellaportas. Gradient-based Adaptive Markov Chain Monte Carlo. In *Advances in Neural Information Processing Systems*, pages 15704–15713, 2019.

[55] 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.

[56] Ziyu Wang, Shakir Mohamed, and Nando Freitas. Adaptive Hamiltonian and Riemann Manifold Monte Carlo. In *International conference on machine learning*, pages 1462–1470. PMLR, 2013.

[57] Changye Wu, Julien Stoehr, and Christian P Robert. Faster Hamiltonian Monte Carlo by learning leapfrog scale. *arXiv preprint arXiv:1810.04449*, 2018.

[58] Kai Xu, Hong Ge, Will Tebbutt, Mohamed Tarek, Martin Trapp, and Zoubin Ghahramani. Advancedhmc. jl: A robust, modular and efficient implementation of advanced HMC algorithms. In *Symposium on Advances in Approximate Bayesian Inference*, pages 1–10. PMLR, 2020.

[59] Kai Xu, Tor Erlend Fjelde, Charles Sutton, and Hong Ge. Couplings for Multinomial Hamiltonian Monte Carlo. In *International Conference on Artificial Intelligence and Statistics*, pages 3646–3654. PMLR, 2021.

[60] Tengchao Yu, Hongqiao Wang, and Jinglai Li. Maximum conditional entropy Hamiltonian Monte Carlo sampler. *SIAM Journal on Scientific Computing*, 2021.

Appendices

A Gradient terms for the adaptation scheme

A.1 Gradients for the entropy approximation

Following the arguments in [13], we can compute the gradient of the term in (13) using

$$\frac{\partial}{\partial \theta_i}\mathcal{L}(\theta) = \operatorname{Tr}\left(\sum_{k=0}^{\infty}(-1)^k \left[D_L\right]^k \frac{\partial}{\partial \theta_i}\left\{D_L\right\}\right) = \mathbb{E}_{N,\varepsilon}\left[\sum_{k=0}^{N}\frac{(-1)^k}{p_k}\varepsilon^\top \left[D_L\right]^k \frac{\partial}{\partial \theta_i}\left\{D_L\right\}\varepsilon\right],$$

which yields a stochastic gradient via a Russian-roulette estimator.

Additionally, to avoid gradients with infinite means even if D_L is not contractive, we consider a spectral normalisation, so that instead of computing recursively $\eta_0 = \varepsilon$ and $\eta_k = D_L \eta_{k-1}$ for $k \in \{1, \dots, N\}$, we set $\bar{\eta}_0 = \varepsilon$ and

$$\bar{\eta}_k = D_L \bar{\eta}_{k-1} \;\cdot\; \min\left\{1, \delta' \left\|\bar{\eta}_{k-1}\right\|_2 / \left\|D_L \bar{\eta}_{k-1}\right\|_2\right\} \tag{15}$$

for $k \in \{1, \dots, N\}$ and $\delta' \in (0,1)$, such as $\delta' = 0.99$ in all our experiments. We obtain an estimator

$$\frac{\partial}{\partial \theta_i}\mathcal{L}(\theta) \approx \mathbb{E}_{N,\varepsilon}\left[\sum_{k=0}^{N}\frac{(-1)^k}{p_k}\bar{\eta}_k^\top \frac{\partial}{\partial \theta_i}\left\{D_L\right\}\varepsilon\right].$$

A.2 Gradients for the penalty function

We used the following penalty function

$$h(x) = (x-\delta)^2 \mathbb{1}_{\{x \in [\delta, \delta_2)\}} + ((\delta_2-\delta)^2 + (\delta_2-\delta)^2(x-\delta_2))\mathbb{1}_{\{x \geqslant \delta_2\}}$$

throughout our experiments with $\delta \in \{0.75, 0.95\}$, and $\delta_2 = 1+\delta$. The motivation was to have a quadratic increase for the penalty term if the largest absolute eigenvalue approaches 1, and then smoothly switch to a linear function for values larger than δ_2. Gradients for this function can be computed routinely using automatic differentiation.

A.3 Gradients for the energy error

We can write the energy error as

$$\begin{aligned}\Delta(q_0, v) &= U(\mathcal{T}_L(v)) - U(q_0) + K(\mathcal{W}_L(v)) - K(C^{-\top}v) \\ &= U\left(q_0 + LhCv - h^2 CC^\top \Xi_L(v) - L\frac{h^2}{2}CC^\top \nabla U(q_0)\right) - U(q_0) \\ &\quad + \frac{1}{2}\left\| v - \frac{h}{2}C\left[\nabla U(q_0) + \nabla U\left(q_L\right)\right] - hC\sum_{\ell=1}^{L-1}\nabla U(q_\ell)\right\|^2 - \frac{1}{2}\left\|v\right\|^2.\end{aligned} \tag{16}$$

Recall from (5) that $\Xi_L(v)$ is a weighted sum of potential energy gradients along the leap-frog trajectory. For computing gradients of the energy-error for the fast approximation, we therefore stop the gradient for all $\nabla U(q_\ell)$ for any $\ell \in \{1, \dots, L\}$.

B Proof of Lemma 1

Proof. We generalise the arguments from [14], Lemma 7. Proceeding by induction over n, we have for the case $n=1$, for any $v \in \mathbb{R}^d$, that $\mathsf{D}\mathcal{T}_1(v) = hC$ and $\mathcal{S}_1(v) = \frac{1}{h}C^{-1}q_0 - \frac{h}{2}C^\top \nabla U(q_0)$ with derivative of zero. For the case $n=2$, using (3) and (5), one obtains

$$\mathsf{D}\mathcal{T}_2(v) - 2hC - h^3 CC^\top \nabla^2 U(\mathcal{T}_1(v))C \tag{17}$$

and moreover

$$\mathsf{D}\mathcal{S}_2(v) = -\frac{h^2}{2} C^\top \nabla^2 U(\mathcal{T}_1(v)) C \tag{18}$$

which establishes (10). Clearly, $\left\|\mathsf{D}\mathcal{S}_2(v)\right\|_2 < \frac{1}{8}$ if $2^2 h^2 < \frac{1}{4\|C^\top \nabla^2 U(\mathcal{T}_1(v))C\|_2}$.
Further, for any $n < L$, again from (3) and (5),

$$\begin{aligned}
\mathsf{D}\mathcal{T}_{n+1}(v) &= (n+1)hC - h^2 CC^\top \mathsf{D}\Xi_{n+1}(v) \\
&= (n+1)hC - h^2 CC^\top \left[\sum_{i=1}^{n} (n+1-i) \nabla^2 U(\mathcal{T}_i(v)) \mathsf{D}\mathcal{T}_i(v)\right] \\
&= (n+1)hC - h^2 CC^\top \left[\sum_{i=1}^{n} (n+1-i) \nabla^2 U(\mathcal{T}_i(v)) ihC \left(\mathrm{I} + \mathsf{D}\mathcal{S}_i(v)\right)\right] \\
&= (n+1)hC + (n+1)hC \left[-h^2 \sum_{i=1}^{n} \frac{(n+1-i)}{n+1} iC^\top \nabla^2 U(\mathcal{T}_i(v)) C \left(\mathrm{I} + \mathsf{D}\mathcal{S}_i(v)\right)\right],
\end{aligned}$$

which shows the representation (10) for the case $n+1$ by recalling that

$$\mathsf{D}\mathcal{T}_{n+1}(v) = (n+1)hC(\mathrm{I} + \mathsf{D}\mathcal{S}_{n+1}(v)).$$

Assume now that $\left\|\mathsf{D}\mathcal{S}_\ell(v)\right\|_2 < 1/8$ holds for all $\ell \leqslant n$. Then for any $v \in \mathbb{R}^d$

$$\begin{aligned}
\left\|\mathsf{D}\mathcal{S}_{n+1}(v)\right\|_2 &\leqslant \frac{h^2}{n+1} \sum_{i=1}^{n} i(n+1-i) \left\|C^\top \nabla^2 U(\mathcal{T}_i(v)) C\right\|_2 \left\|\mathrm{I} + \mathsf{D}\mathcal{S}_i(v)\right\|_2 \\
&\leqslant \frac{h^2}{n+1} \sum_{i=1}^{n} \frac{L^2}{4} \left\|C^\top \nabla^2 U(\mathcal{T}_i(v)) C\right\|_2 \left\|\mathrm{I} + \mathsf{D}\mathcal{S}_i(v)\right\|_2 \\
&\leqslant \frac{h^2}{n+1} \sum_{i=1}^{n} \frac{L^2}{4} \frac{1}{4L^2h^2} \left(1 + \frac{1}{8}\right) \leqslant \frac{1}{8}
\end{aligned}$$

where the second inequality follows from $(n+1-i)i \leqslant (\frac{n+1-i+i}{2})^2 \leqslant \frac{L^2}{4}$, whereas the third inequality follows from the induction hypothesis and the assumption $L^2 h^2 < \sup_q \frac{1}{4\|C^\top \nabla^2 U(q) C\|_2}$. $\square$

C Extension to learn non-linear transformations

The suggested approach can perform poorly for non-convex potentials or even convex potentials such as arsing in a logistic regression model for some data sets. We illustrate here how to learn a reasonable proposal for a general potential function by considering some version of position-dependent preconditioning. Consider an invertible differentiable transformation $f : \mathbb{R}^d \to \mathbb{R}^d$. The idea now is to run HMC with unit mass matrix for the transformed variables $z = f^{-1}(q)$ where $q \sim \pi$. Write $\tilde{\pi}$ for the density of z and let $\tilde{U}$ be the corresponding potential energy function which is given by

$$\tilde{U}(z) = U(H(z)) - \log\left|\det \mathsf{D}f(z)\right|$$

with gradient

$$\nabla\tilde{U}(z) = \mathsf{D}f(z)^\top \nabla U(f(z)) - \nabla \log\left|\det \mathsf{D}f(z)\right|.$$

The transformation f as well as $\tilde{U}$ generally depend on some parameters θ that we again omit for a less convoluted notation. Our approach can be seen as an alternative for instance to [31] where such a transformation is first learned by trying to approximate $\tilde{\pi}$ with a standard Gaussian density using variational inference, while the HMC hyperparameters are adapted in a second step using Bayesian optimisation.

We write $\tilde{\mathcal{T}}_L : v \mapsto z_L$ for the transformation that maps the initial velocity $v = p_0 \sim \mathcal{N}(0, \mathrm{I})$ to the L-th leapfrog step z_L, starting at z_0 based on the potential function $\tilde{U}$ with unit mass matrix $M = \mathrm{I}$. Analogously, we define the mapping $\tilde{\mathcal{W}}_L : v \mapsto p_L$ and similarly to (7), we define

$$\tilde{\mathcal{S}}_L(v) = \frac{1}{Lh}\tilde{\mathcal{T}}_L(v) - v.$$

We can then reparametrize the proposal at the point $q_0 = f(z_0)$ by $v \mapsto f(\tilde{\mathcal{T}}_L(v))$. Consequently, the log-density of the proposal is given by

$$\log r_L(f(\tilde{\mathcal{T}}_L(v))) = \log \nu(v) - \log|\det \mathsf{D} f(\tilde{\mathcal{T}}_L(v))| - \log|\det \mathsf{D}\tilde{\mathcal{T}}_L(v)|,$$

and we can write

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

We use the same approximation

$$\mathsf{D}\tilde{\mathcal{S}}_L(v) \approx -h^2 \frac{L^2-1}{6}\, \nabla^2 \tilde{U}(z_{\lfloor L/2 \rfloor})$$

based on the transformed Hessian now.

Hessian-vector products can similarly be computed using vector-Jacobian products: With $g(z) = \texttt{grad}(\tilde{U}, z)$, we then compute $\nabla^2 \tilde{U}(z) w = \texttt{vjp}(g, z, w)^\top$ for $z = f^{-1}(\texttt{stop\_grad}(f(z_{\lfloor L/2 \rfloor})))$. The motivation for stopping the gradients comes from considering the special case $f \colon z \mapsto Cz$ that corresponds to the position-independent preconditioning scheme above. For such a linear transformation,

$$\tilde{U}(z) = C^\top \nabla^2 U(Cz) C.$$

To recover the previous case, we stop gradients at $q_{\lfloor L/2 \rfloor} = f(z_{\lfloor L/2 \rfloor}) = C z_{\lfloor L/2 \rfloor}$.

Gradients for the log-accept ratio can be computed based on the log-accept ratio of the transformed chain [35]. The energy error of the transformed chain is

$$\begin{aligned}
\Delta_\theta(q_0, v) =& U_\theta(\tilde{\mathcal{T}}_L(v)) - U_\theta(f^{-1}(q_0)) + K(\tilde{\mathcal{W}}_L(v)) - K(v) \\
=& U\Big\{ f\Big[f^{-1}(q_0) + Lhv - h^2 \tilde{\Xi}_L(v) \\
& \qquad - L\frac{h^2}{2}\left(\mathsf{D} f(f^{-1}(q_0))^\top \nabla U(q_0) - \nabla \log|\det \mathsf{D} f(f^{-1}(q_0))| \right) \Big] \Big\} \\
& + \log|\det \mathsf{D} f(z_L)| - U(q) + \log|\det \mathsf{D} f(f^{-1}(q))| \\
& + \frac{1}{2}\Bigg\| v - \frac{h}{2}\big[\mathsf{D} f(z_0)^\top \nabla U(f(z_0)) - \nabla \log|\det \mathsf{D} f(z_0) + \mathsf{D} f(z_L)^\top \nabla U(f(z_L)) \\
& \qquad - \nabla \log|\det \mathsf{D} f(z_L)| \big] - h \sum_{\ell=1}^{L-1} \mathsf{D} f(z_\ell)^\top \nabla U(f(z_\ell)) - \nabla \log|\det \mathsf{D} f(z_\ell)| \Bigg\|^2 \\
& - \frac{1}{2}\|v\|^2,
\end{aligned}$$

where

$$\tilde{\Xi}_L(v) = \sum_{i=1}^{L} (L-i)\left[\mathsf{D} f(z_i)^\top \nabla U(f(z_i)) - \nabla \log|\det \mathsf{D} f(z_i)| \right]$$

and $z_0, \ldots, z_L$ is the leap-frog trajectory starting at $z_0 = f^{-1}(q_0)$. We also stop all U gradients, *i.e.* $\nabla U(f(z_\ell)) \leftarrow \texttt{stop\_grad}(\nabla U(f(z_\ell)))$. It can be seen that this recovers the above setting if $f \colon z \mapsto Cz$.

D Gradient-based adaptation using the expected squared jumping distance and variations

We consider the different loss functions

$$\mathcal{F}_{\text{GSM}}(\theta) = -\int\int \pi(q_0)\nu(v)\Big[\log a\{(q_0,v),(\mathcal{T}_L(v),\mathcal{W}_L(v))\} - \beta \log r_L(\mathcal{T}_L(v))\Big]\mathrm{d}v\mathrm{d}q_0 \tag{19}$$

$$\mathcal{F}_{\text{ESJD}}(\theta) = -\int\int \pi(q_0)\nu(v)\Big[a\{(q_0,v),(\mathcal{T}_L(v),\mathcal{W}_L(v))\}\,\|q_0-\mathcal{T}_L(v)\|^2\Big]\mathrm{d}v\mathrm{d}q_0 \tag{20}$$

$$\begin{aligned}\mathcal{F}_{\text{L2HMC}}(\theta) = -\int\int \pi(q_0)\nu(v)\Big[&\frac{a\{(q_0,v),(\mathcal{T}_L(v),\mathcal{W}_L(v))\}\,\|q_0-\mathcal{T}_L(v)\|^2}{\lambda} \\ &- \frac{\lambda}{a\{(q_0,v),(\mathcal{T}_L(v),\mathcal{W}_L(v))\}\,\|q_0-\mathcal{T}_L(v)\|^2}\Big]\mathrm{d}v\mathrm{d}q_0.\end{aligned} \tag{21}$$

The L2HMC objective (21) has been suggested in Levy et al. [40] for learning generalisations of HMC, although we ignore a burn-in term that has been included originally. In our implementation, we adapt $\lambda > 0$ online as a moving average of the expected squared jumping distance. The objectives (20) and (21) can be optimized using stochastic gradient descent similar to Algorithm 1 without the approximations as required for the GSM objective (19).

E Proof of the HMC proposal reparameterizations

For completeness, we provide a proof of the reparameterization (3) and (4) of the L-th step position q_L and momentum p_L using the velocity v that relates to the initial momentum $p_0 \sim \mathcal{N}(0, M)$ via $p_0 = C^{-\top}v$. Such representations with an identity mass matrix have been used previously in [42, 21, 14].

Proof. We proceed by induction over $\ell \in \{1, \dots, L\}$. For the case $\ell = 1$, the recursions in (2) imply

$$q_1 = q_0 + hCC^\top\left[p_0 - \frac{h}{2}\nabla U(q_0)\right] = q_0 + hCv - \frac{h}{2}CC^\top\nabla U(q_0)$$

and

$$p_1 = \left[p_0 - \frac{h}{2}\nabla U(q_0)\right] - \frac{h}{2}\nabla U(q_1) = C^{-\top}v - \frac{h}{2}\left[\nabla U(q_0) + \nabla U(q_1)\right].$$

Suppose now that the representations hold for $1 \leqslant \ell < L$. Then, using the recursions in (2),

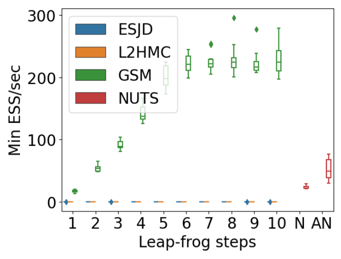
$$\begin{aligned}q_{\ell+1} &= q_\ell + hCC^\top\left[p_\ell - \frac{h}{2}\nabla U(q_\ell)\right] \\ &= q_0 - \left[\frac{\ell h^2}{2}CC^\top + \frac{h}{2}CC^\top\right]\nabla U(q_0) + \left[\ell hC + hCC^\top C^{-\top}\right]v - h^2CC^\top\nabla U(q_\ell) \\ &\quad - h^2CC^\top\sum_{i=1}^{\ell-1}\nabla U(q_i) - h^2CC^\top\Xi_\ell(v) \\ &= q_0 - \left[(\ell+1)\frac{h^2}{2}CC^\top\right]\nabla U(q_0) + (\ell+1)hCv - h^2CC^\top\sum_{i=1}^{\ell}\nabla(\ell+1-i)\nabla U(q_i).\end{aligned}$$

This establishes the representation for q_L. The induction step for the momentum is a straightforward application of (2) to the induction hypothesis.

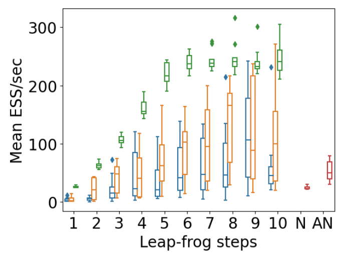
□

F Gaussian targets experiments

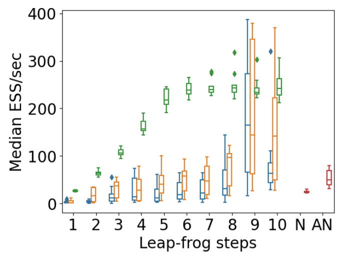
F.1 High-dimensional Gaussian targets



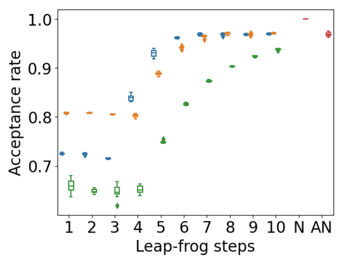
(a)



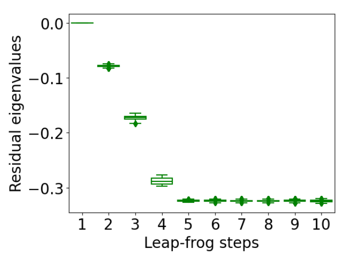
(b)



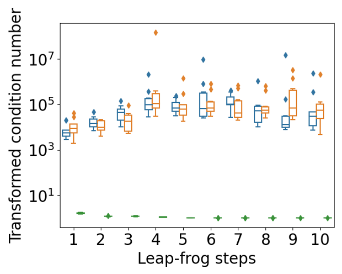
(c)



(d)

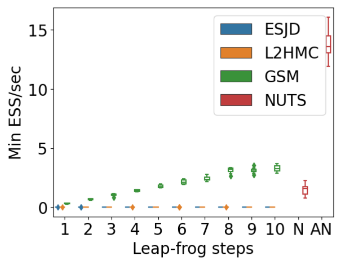


(e)

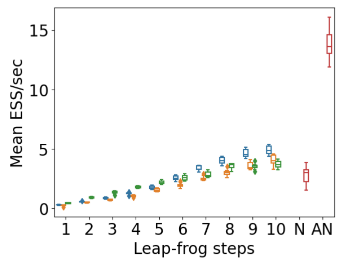


(f)

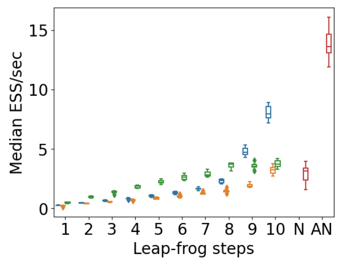
Figure 7: Anisotropic Gaussian target ($d = 1000$). Minimum (7a), mean (7b) and median (7c) effective sample size of $q \mapsto q_i$ per second. Average acceptance rates in 7d and estimates of the eigenvalues of D_L in 7e. Condition number of transformed Hessian $C^\top \Sigma^{-1} C$ in 7f.



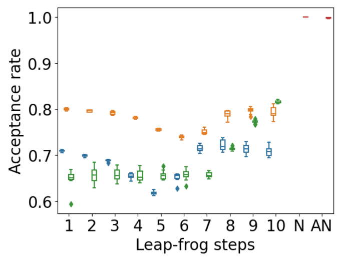
(a)



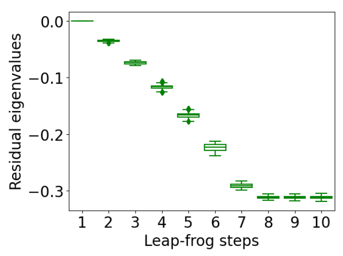
(b)



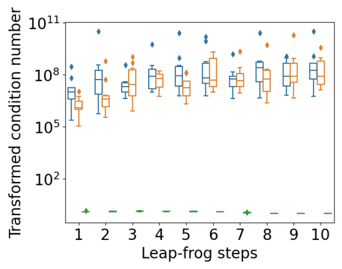
(c)



(d)



(e)



(f)

Figure 8: Independent Gaussian target ($d = 10000$). Minimum (8a), mean (8b) and median (8c) effective sample size of $q \mapsto q_i$ per second. Average acceptance rates in 8d and estimates of the eigenvalues of D_L in 8e. Condition number of transformed Hessian $C^\top \Sigma^{-1} C$ in 8f.

F.2 Ill-conditioned anisotropic Gaussian target

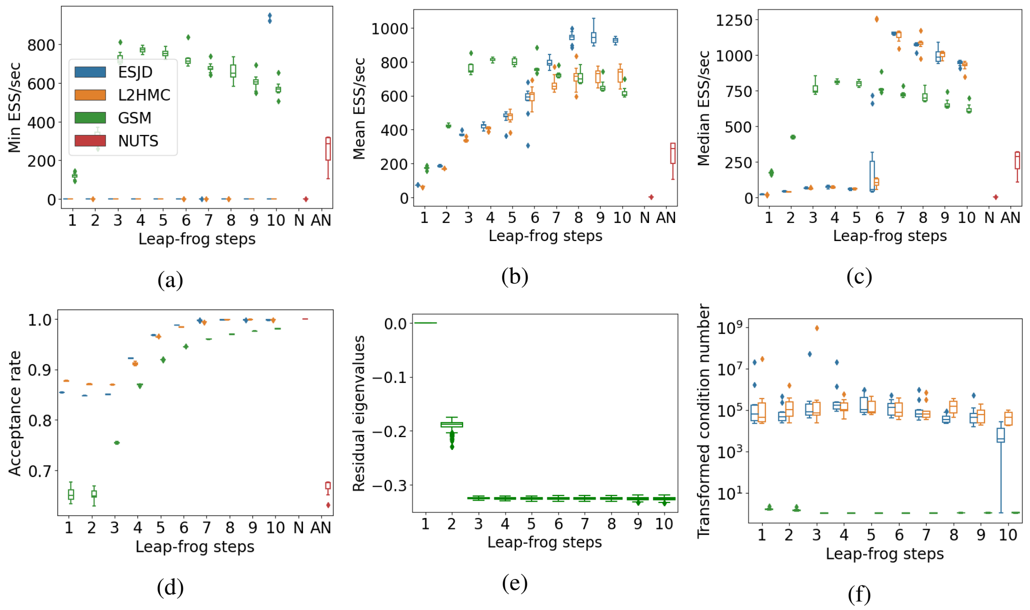


Figure 9: Ill-conditioned Gaussian target ($d = 100$). Minimum (9a), mean (9b) and median (9c) effective sample size of $q \mapsto q_i$ per second. Average acceptance rates in 9d and estimates of the eigenvalues of D_L using power iteration in 9e. Condition number of transformed Hessian $C^\top \Sigma^{-1} C$ in 9f. Values computed after adaptation.

F.3 Correlated Gaussian target

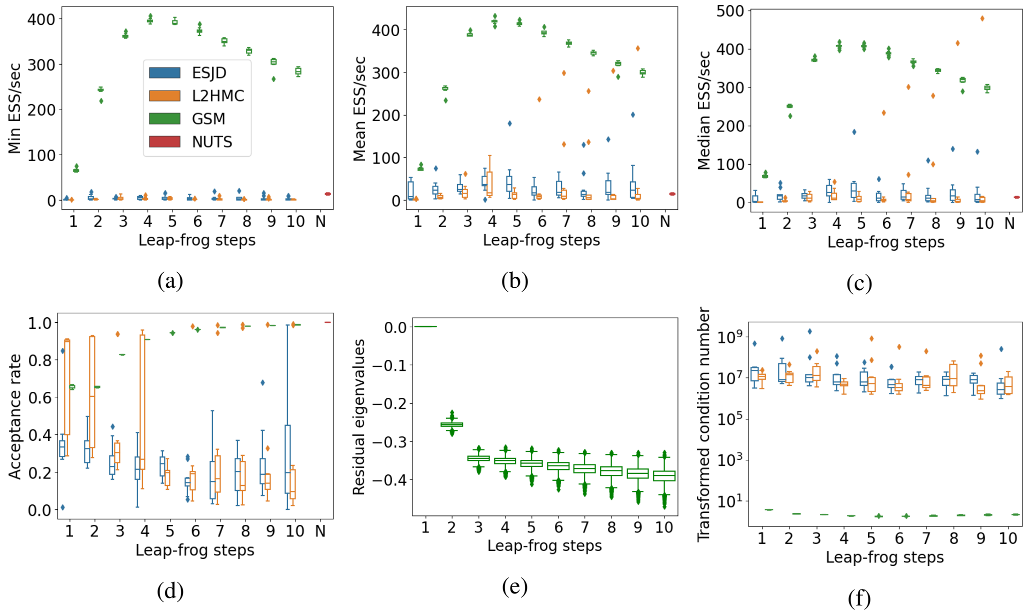


Figure 10: Correlated Gaussian target ($d = 51$). Minimum (10a), mean (10b) and median (10c) effective sample size of $q \mapsto q_i$ per second. Average acceptance rates in 10d and estimates of the eigenvalues of D_L using power iteration in 10e. Condition number of transformed Hessian $C^\top \Sigma^{-1} C$ in 10f. Values computed after adaptation.

F.4 IID Gaussian target

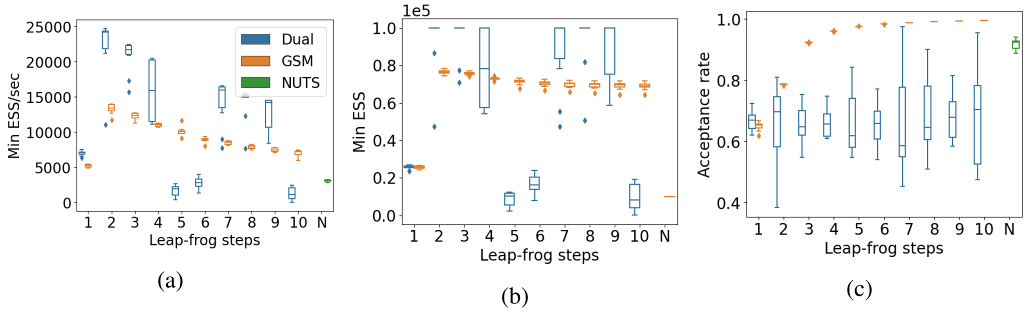


Figure 11: IID Gaussian target ($d = 10$). Minimum effective sample size of $q \mapsto q_i$ per second in 11a and absolute minimum effective sample size where NUTS is run for $1/10$-th of the iterations of the other schemes in 11b. Average acceptance rates in 11c. Values computed after adaptation.

G Logistic regression experiments

G.1 Australian credit data

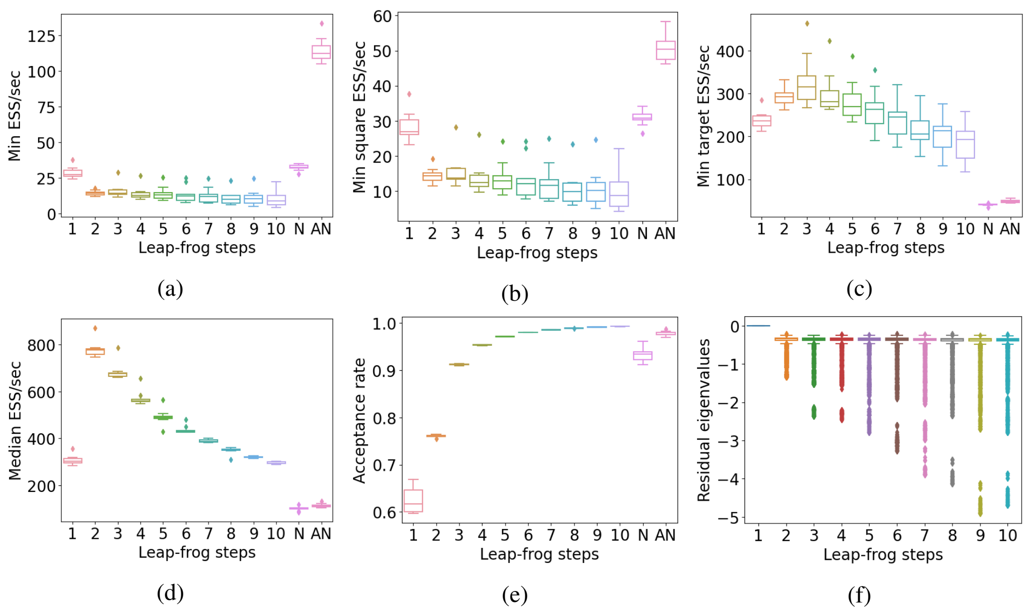
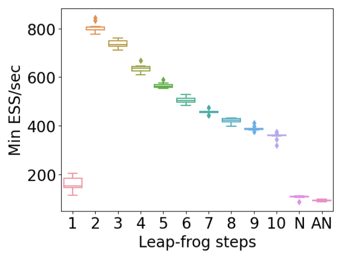
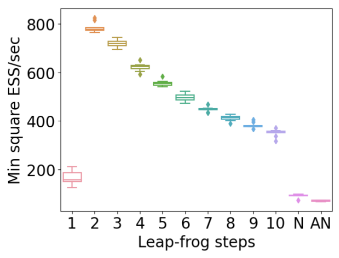


Figure 12: Bayesian logistic regression for Australian Credit data set ($d = 15$). Minimum effective sample size per second after adaptation of $q \mapsto q_i$ in 12a, of $q \mapsto q_i^2$ in 12b and of $q \mapsto \log \pi(q)$ in 12b. Median marginal effective sample per second in 12d and average acceptance rates in 12e and estimates of the eigenvalues of D_L in 12f.

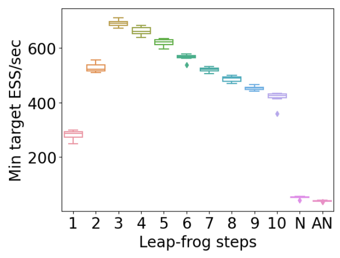
G.2 Heart data



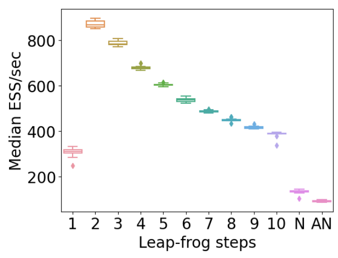
(a)



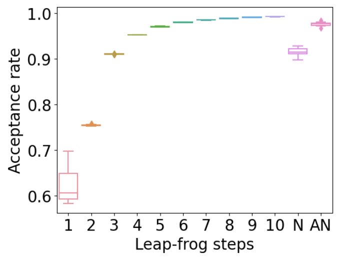
(b)



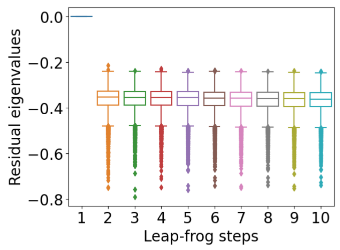
(c)



(d)



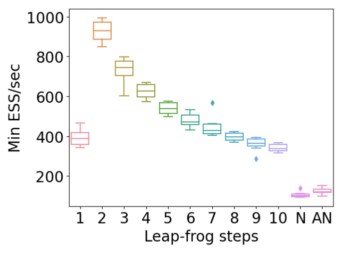
(e)



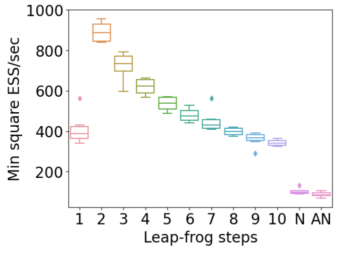
(f)

Figure 13: Bayesian logistic regression for heart data set ($d = 14$). Minimum effective sample size per second after adaptation of $q \mapsto q_i$ in 13a, of $q \mapsto q_i^2$ in 13b and of $q \mapsto \log \pi(q)$ in 13b. Median marginal effective sample per second in 13d and average acceptance rates in 13e and estimates of the eigenvalues of D_L in 13f.

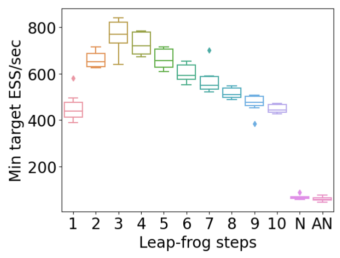
G.3 Pima data



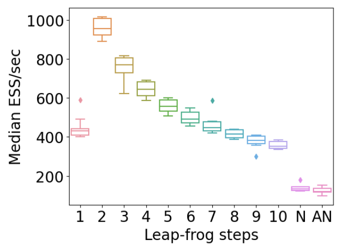
(a)



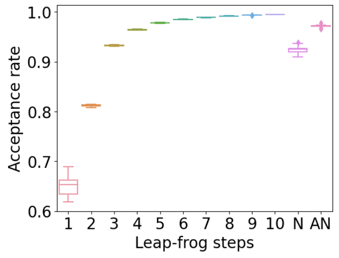
(b)



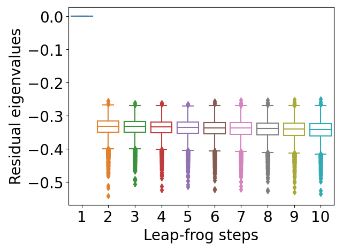
(c)



(d)



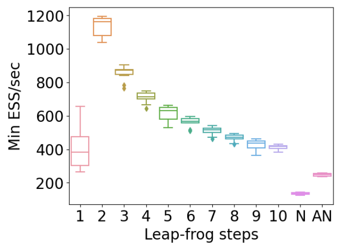
(e)



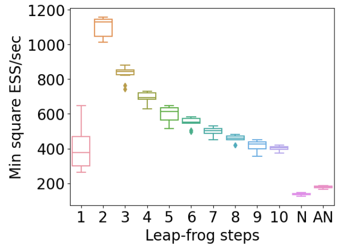
(f)

Figure 14: Bayesian logistic regression for Pima data set ($d = 8$). Minimum effective sample size per second after adaptation of $q \mapsto q_i$ in 14a, of $q \mapsto q_i^2$ in 14b and of $q \mapsto \log \pi(q)$ in 14c. Median marginal effective sample per second in 14d and average acceptance rates in 14e and estimates of the eigenvalues of D_L in 14f.

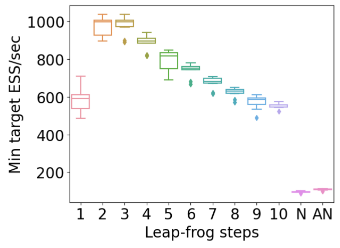
G.4 Ripley data



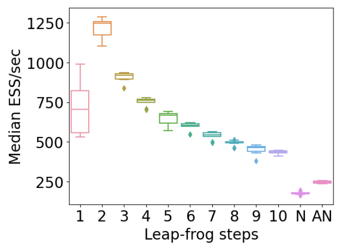
(a)



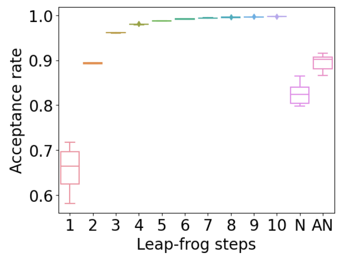
(b)



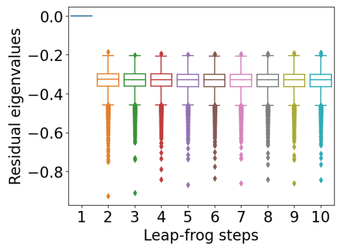
(c)



(d)



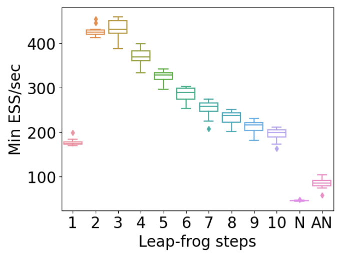
(e)



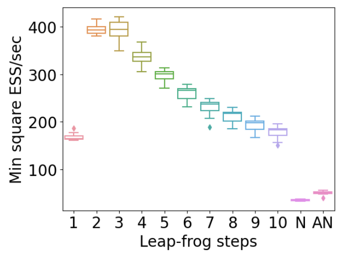
(f)

Figure 15: Bayesian logistic regression for Ripley data set ($d = 3$). Minimum effective sample size per second after adaptation of $q \mapsto q_i$ in 15a, of $q \mapsto q_i^2$ in 15b and of $q \mapsto \log \pi(q)$ in 15c. Median marginal effective sample per second in 15d and average acceptance rates in 15e and estimates of the eigenvalues of D_L in 15f.

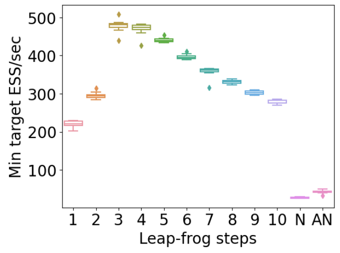
G.5 German credit data



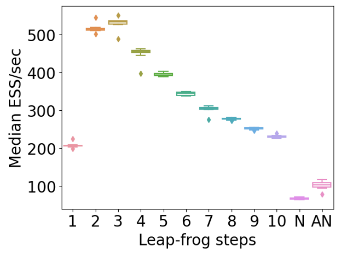
(a)



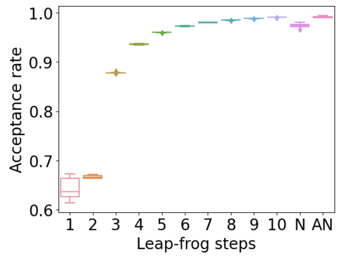
(b)



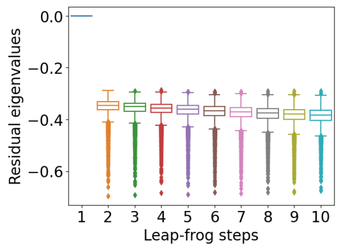
(c)



(d)



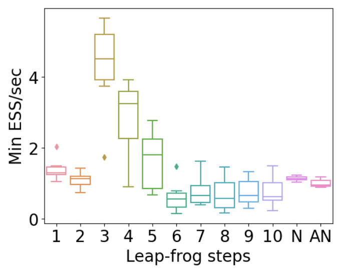
(e)



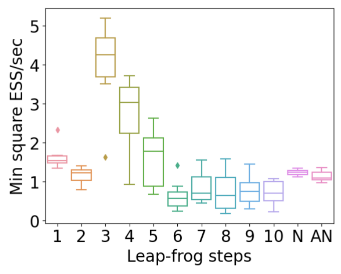
(f)

Figure 16: Bayesian logistic regression for German credit data set ($d = 25$). Minimum effective sample size per second after adaptation of $q \mapsto q_i$ in 16a, of $q \mapsto q_i^2$ in 16b and of $q \mapsto \log \pi(q)$ in 16c. Median marginal effective sample per second in 16d and average acceptance rates in 16e and estimates of the eigenvalues of D_L in 16f.

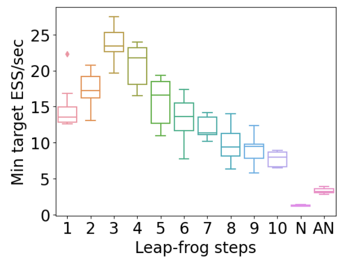
G.6 Caravan data



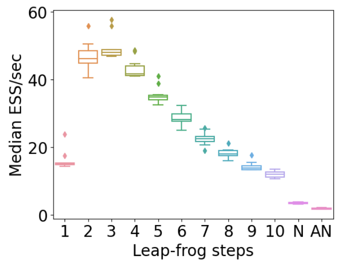
(a)



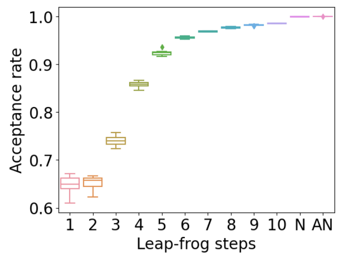
(b)



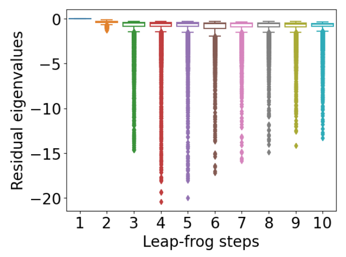
(c)



(d)



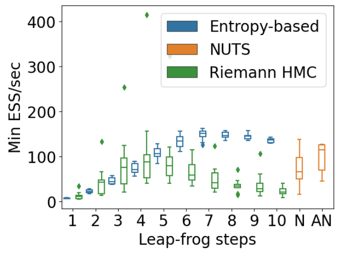
(e)



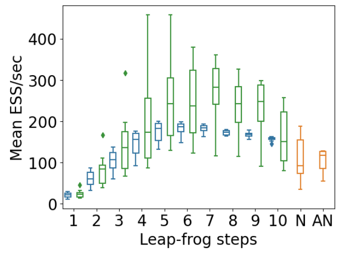
(f)

Figure 17: Bayesian logistic regression for Caravan data set ($d = 87$). Minimum effective sample size per second after adaptation of $q \mapsto q_i$ in 17a, of $q \mapsto q_i^2$ in 17b and of $q \mapsto \log \pi(q)$ in 17c. Median marginal effective sample per second in 17d and average acceptance rates in 17e and estimates of the eigenvalues of D_L in 17f.

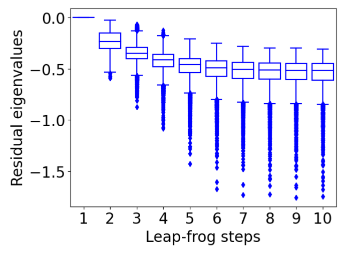
H Log-Gaussian Cox Point Process



(a)



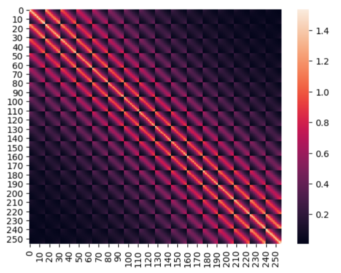
(b)

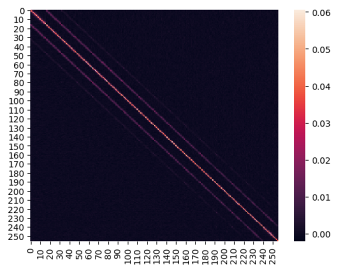


(c)

Figure 18: Cox process in dimension $d = 64$. Minimum (18a) and mean (18b) effective sample size per second after adaptation. Estimates of the eigenvalues of D_L using power iteration in (18c).

I Stochastic volatility model


(a) Inverse mass matrix $(\Lambda + \Sigma^{-1})^{-1}$ of the Riemann manifold based samplers.


(b) Inverse mass matrix $CC^\top$ for the entropy-based scheme with $L = 1$.

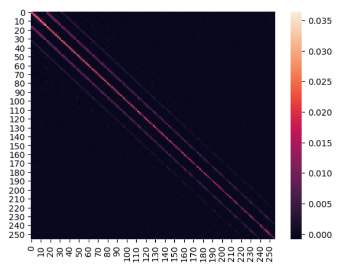
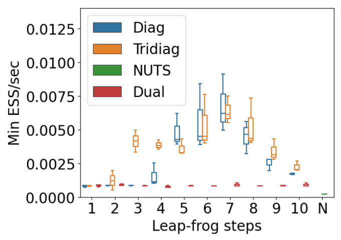
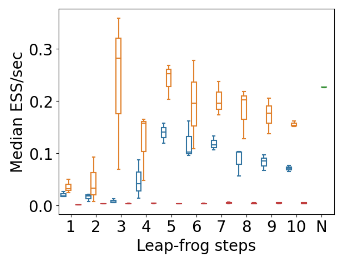

(c) Inverse mass matrix $CC^\top$ for the entropy-based scheme with $L = 5$.

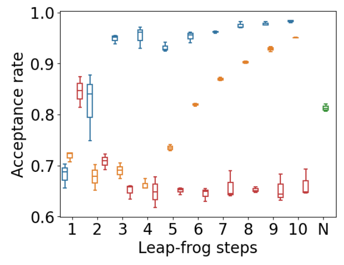
Figure 19: Inverse mass matrices for the Cox process with $d = 256$ for the different schemes.



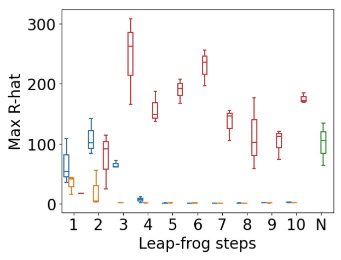
(a)



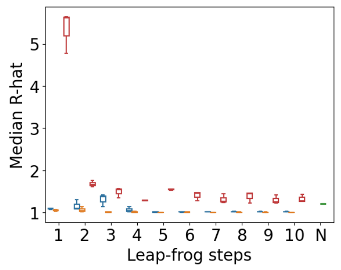
(b)



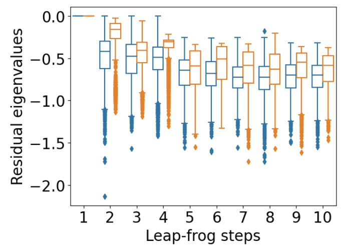
(c)



(d)

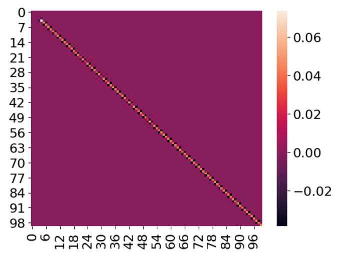


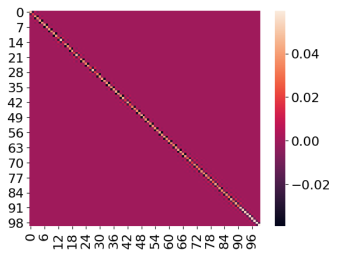
(e)



(f)

Figure 20: MCMC mixing efficiency for the stochastic volatility model ($d = 2519$) after adaptation: Minimum (20a) and median (20b) effective sample size per second. Maximum (20d) and median (20e) $\hat{R}$ of $q \mapsto q_i$. Average acceptance rates (20c) and estimates of the eigenvalues of D_L (20f).


(a) First 100 dimensions of M^{-1} for $L = 5$ with a tridiagonal mass matrix.


(b) Last 100 dimensions of M^{-1} for $L = 5$ with a tridiagonal mass matrix.

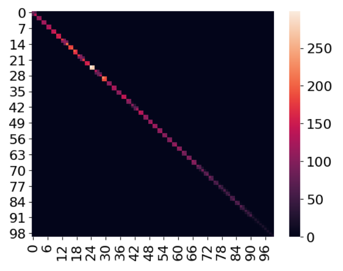

(c) Last 100 dimensions of M for $L = 5$ with a tridiagonal mass matrix.

Figure 21: Learned (inverse) mass matrices for the stochastic volatility model.