# Gradient-free Hamiltonian Monte Carlo with Efficient Kernel Exponential Families

**Heiko Strathmann**[*] **Dino Sejdinovic**[+] **Samuel Livingstone**[o] **Zoltan Szabo**[*] **Arthur Gretton**[*]

[*]Gatsby Unit
University College London

[+]Department of Statistics
University of Oxford

[o]School of Mathematics
University of Bristol

## Abstract

We propose *Kernel Hamiltonian Monte Carlo (KMC)*, a gradient-free adaptive MCMC algorithm based on Hamiltonian Monte Carlo (HMC). On target densities where classical HMC is not an option due to intractable gradients, KMC adaptively learns the target's gradient structure by fitting an exponential family model in a Reproducing Kernel Hilbert Space. Computational costs are reduced by two novel efficient approximations to this gradient. While being asymptotically exact, KMC mimics HMC in terms of sampling efficiency, and offers substantial mixing improvements over state-of-the-art gradient free samplers. We support our claims with experimental studies on both toy and real-world applications, including Approximate Bayesian Computation and exact-approximate MCMC.

## 1 Introduction

Estimating expectations using Markov Chain Monte Carlo (MCMC) is a fundamental approximate inference technique in Bayesian statistics. MCMC itself can be computationally demanding, and the expected estimation error depends directly on the correlation between successive points in the Markov chain. Therefore, efficiency can be achieved by taking large steps with high probability.

Hamiltonian Monte Carlo [1] is an MCMC algorithm that improves efficiency by exploiting gradient information. It simulates particle movement along the contour lines of a dynamical system constructed from the target density. Projections of these trajectories cover wide parts of the target's support, and the probability of accepting a move along a trajectory is often close to one. Remarkably, this property is mostly invariant to growing dimensionality, and HMC here often is superior to random walk methods, which need to decrease their step size at a much faster rate [1, Sec. 4.4].

Unfortunately, for a large class of problems, gradient information is not available. For example, in Pseudo-Marginal MCMC (PM-MCMC) [2, 3], the posterior does not have an analytic expression, but can only be estimated at any given point, e.g. in Bayesian Gaussian Process classification [4]. A related setting is MCMC for Approximate Bayesian Computation (ABC-MCMC), where the posterior is approximated through repeated simulation from a likelihood model [5, 6]. In both cases, HMC cannot be applied, leaving random walk methods as the only mature alternative. There have been efforts to mimic HMC's behaviour using stochastic gradients from mini-batches in Big Data [7], or stochastic finite differences in ABC [8]. Stochastic gradient based HMC methods, however, often suffer from low acceptance rates or additional bias that is hard to quantify [9].

Random walk methods can be tuned by matching scaling of steps and target. For example, Adaptive Metropolis-Hastings (AMH) [10, 11] is based on learning the global scaling of the target from the history of the Markov chain. Yet, for densities with nonlinear support, this approach does not work very well. Recently, [12] introduced a Kernel Adaptive Metropolis-Hastings (KAMH) algorithm whose proposals are locally aligned to the target. By adaptively learning target covariance in a Reproducing Kernel Hilbert Space (RKHS), KAMH achieves improved sampling efficiency.

In this paper, we extend the idea of using kernel methods to learn efficient proposal distributions [12]. Rather than *locally* smoothing the target density, however, we estimate its gradients *globally*. More precisely, we fit an infinite dimensional exponential family model in an RKHS via score matching [13, 14]. This is a non-parametric method of modelling the log unnormalised target density as an RKHS function, and has been shown to approximate a rich class of density functions arbitrarily well. More importantly, the method has been empirically observed to be relatively robust to increasing dimensionality – in sharp contrast to classical kernel density estimation [15, Sec. 6.5]. Gaussian Processes (GP) were also used in [16] as an emulator of the target density in order to speed up HMC, however, this requires access to the target in closed form, to provide training points for the GP.

We require our adaptive KMC algorithm to be computationally efficient, as it deals with high-dimensional MCMC chains of growing length. We develop two novel approximations to the infinite dimensional exponential family model. The first approximation, *score matching lite*, is based on computing the solution in terms of a lower dimensional, yet growing, subspace in the RKHS. KMC with score matching lite (*KMC lite*) is geometrically ergodic on the same class of targets as standard random walks. The second approximation uses a finite dimensional feature space (*KMC finite*), combined with random Fourier features [17]. KMC finite is an efficient online estimator that allows to use *all* of the Markov chain history, at the cost of decreased efficiency in unexplored regions. A choice between KMC lite and KMC finite ultimately depends on the ability to initialise the sampler within high-density regions of the target; alternatively, the two approaches could be combined.

Experiments show that KMC inherits the efficiency of HMC, and therefore mixes significantly better than state-of-the-art gradient-free adaptive samplers on a number of target densities, including on synthetic examples, and when used in PM-MCMC and ABC-MCMC. All code can be found at `https://github.com/karlnapf/kernel_hmc`

## 2 Background and Previous Work

Let the domain of interest $\mathcal{X}$ be a compact[1] subset of $\mathbb{R}^d$, and denote the unnormalised *target* density on $\mathcal{X}$ by $\pi$. We are interested in constructing a Markov chain $x_1 \to x_2 \to \dots$ such that $\lim_{t\to\infty} x_t \sim \pi$. By running the Markov chain for a long time $T$, we can consistently approximate any expectation w.r.t. $\pi$. Markov chains are constructed using the Metropolis-Hastings algorithm, which at the current state $x_t$ draws a point from a proposal mechanism $x^* \sim Q(\cdot|x_t)$, and sets $x_{t+1} \leftarrow x^*$ with probability $\min(1, [\pi(x^*)Q(x_t|x^*)]/[\pi(x_t)Q(x^*|x_t)])$, and $x_{t+1} \leftarrow x_t$ otherwise. We assume that $\pi$ is intractable,[2] i.e. that we can neither evaluate $\pi(x)$ nor[3] $\nabla \log \pi(x)$ for any $x$, but can only estimate it unbiasedly via $\hat{\pi}(x)$. Replacing $\pi(x)$ with $\hat{\pi}(x)$ results in PM-MCMC [2, 3], which asymptotically remains exact (*exact-approximate inference)*.

**(Kernel) Adaptive Metropolis-Hastings** In the absence of $\nabla \log \pi$, the usual choice of $Q$ is a random walk, i.e. $Q(\cdot|x_t) = \mathcal{N}(\cdot|x_t, \Sigma_t)$. A popular choice of the scaling is $\Sigma_t \propto I$. When the scale of the target density is not uniform across dimensions, or if there are strong correlations, the AMH algorithm [10, 11] improves mixing by adaptively learning global covariance structure of $\pi$ from the history of the Markov chain. For cases where the local scaling does not match the global covariance of $\pi$, i.e. the support of the target is nonlinear, KAMH [12] improves mixing by learning the target covariance in a RKHS. KAMH proposals are Gaussian with a covariance that matches the local covariance of $\pi$ around the current state $x_t$, without requiring access to $\nabla \log \pi$.

**Hamiltonian Monte Carlo** Hamiltonian Monte Carlo (HMC) uses deterministic, measure-preserving maps to generate efficient Markov transitions [1, 18]. Starting from the negative log target, referred to as the *potential energy* $U(q) = -\log \pi(q)$, we introduce an auxiliary *momentum* variable $p \sim \exp(-K(p))$ with $p \in \mathcal{X}$. The joint distribution of $(p, q)$ is then proportional to $\exp(-H(p, q))$, where $H(p, q) := K(p) + U(q)$ is called the *Hamiltonian*. $H(p, q)$ defines a *Hamiltonian flow*, parametrised by a trajectory length $t \in \mathbb{R}$, which is a map $\phi_t^H : (p, q) \mapsto (p^*, q^*)$ for which $H(p^*, q^*) = H(p, q)$. This allows constructing $\pi$-invariant Markov chains: for a chain at state $q = x_t$, repeatedly (i) re-sample $p' \sim \exp(-K(\cdot))$, and then (ii) apply the Hamiltonian flow

[1] The compactness restriction is imposed to satisfy the assumptions in [13].

[2] $\pi$ is analytically intractable, as opposed to computationally expensive in the Big Data context.

[3] Throughout the paper $\nabla$ denotes the gradient operator w.r.t. to $x$.

for time $t$, giving $(p^*, q^*) = \phi_t^H(p', q)$. The flow can be generated by the *Hamiltonian operator*

$$\frac{\partial K}{\partial p}\frac{\partial}{\partial q} - \frac{\partial U}{\partial q}\frac{\partial}{\partial p} \tag{1}$$

In practice, (1) is usually unavailable and we need to resort to approximations. Here, we limit ourselves to the leap-frog integrator; see [1] for details. To correct for discretisation error, a Metropolis acceptance procedure can be applied: starting from $(p', q)$, the end-point of the approximate trajectory is accepted with probability $\min\left[1, \exp\left(-H(p^*, q^*) + H(q, p')\right)\right]$. HMC is often able to propose distant, uncorrelated moves with a high acceptance probability.

**Intractable densities** In many cases the gradient of $\log \pi(q) = -U(q)$ cannot be written in closed form, leaving random-walk based methods as the state-of-the-art [12, 11]. We aim to overcome random-walk behaviour, so as to obtain significantly more efficient sampling [1].

## 3 Kernel Induced Hamiltonian Dynamics

KMC replaces the potential energy in (1) by a kernel induced surrogate computed from the history of the Markov chain. This surrogate does not require gradients of the log-target density. The surrogate induces a kernel Hamiltonian flow, which can be numerically simulated using standard leap-frog integration. As with the discretisation error in HMC, any deviation of the kernel induced flow from the true flow is corrected via a Metropolis acceptance procedure. This here also contains the estimation noise from $\hat{\pi}$ and re-uses previous values of $\hat{\pi}$, c.f. [3, Table 1]. Consequently, the stationary distribution of the chain remains correct, given that we take care when adapting the surrogate.

**Infinite Dimensional Exponential Families in a RKHS** We construct a kernel induced potential energy surrogate whose gradients approximate the gradients of the true potential energy $U$ in (1), without accessing $\pi$ or $\nabla\pi$ directly, but only using the history of the Markov chain. To that end, we model the (unnormalised) target density $\pi(x)$ with an infinite dimensional exponential family model [13] of the form

$$\text{const} \times \pi(x) \approx \exp\left(\langle f, k(x, \cdot)\rangle_{\mathcal{H}} - A(f)\right), \tag{2}$$

which in particular implies $\nabla f \approx -\nabla U = \nabla \log \pi$. Here $\mathcal{H}$ is a RKHS of real valued functions on $\mathcal{X}$. The RKHS has a uniquely associated symmetric, positive definite *kernel* $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, which satisfies $f(x) = \langle f, k(x, \cdot)\rangle_{\mathcal{H}}$ for any $f \in \mathcal{H}$ [19]. The canonical feature map $k(\cdot, x) \in \mathcal{H}$ here takes the role of the *sufficient statistics* while $f \in \mathcal{H}$ are the *natural parameters*, and $A(f) := \log \int_{\mathcal{X}} \exp(\langle f, k(x, \cdot)\rangle_{\mathcal{H}}) dx$ is the cumulant generating function.Eq. (2) defines broad class of densities: when universal kernels are used, the family is dense in the space of continuous densities on compact domains, with respect to e.g. Total Variation and KL [13, Section 3]. It is possible to consistently fit an *unnormalised* version of (2) by directly minimising the expected gradient mismatch between the model (2) and the true target density $\pi$ (observed through the Markov chain history). This is achieved by generalising the score matching approach [14] to infinite dimensional parameter spaces. The technique avoids the problem of dealing with the intractable $A(f)$, and reduces the problem to solving a linear system. More importantly, the approach is observed to be relatively robust to increasing dimensions. We return to estimation in Section 4, where we develop two efficient approximations. For now, assume access to an $\hat{f} \in \mathcal{H}$ such that $\nabla f(x) \approx \nabla \log \pi(x)$.

**Kernel Induced Hamiltonian Flow** We define a kernel induced Hamiltonian operator by replacing $U$ in the potential energy part $\frac{\partial U}{\partial p}\frac{\partial}{\partial q}$ in (1) by our kernel surrogate $U_k = f$. It is clear that, depending on $U_k$, the resulting kernel induced Hamiltonian flow differs from the original one. That said, any bias on the resulting Markov chain, in addition to discretisation error from the leap-frog integrator, is naturally corrected for in the Pseudo-Marginal Metropolis step. We accept an end-point $\phi_t^{H_k}(p', q)$ of a trajectory starting at $(p', q)$ along the *kernel induced* flow with probability

$$\min\left[1, \exp\left(-H\left(\phi_t^{H_k}(p', q)\right) + H(p', q)\right)\right], \tag{3}$$

where $H\left(\phi_t^{H_k}(p', q)\right)$ corresponds to the *true* Hamiltonian at $\phi_t^{H_k}(p', q)$. Here, in the Pseudo-Marginal context, we replace both terms in the ratio in (3) by unbiased estimates, i.e., we replace

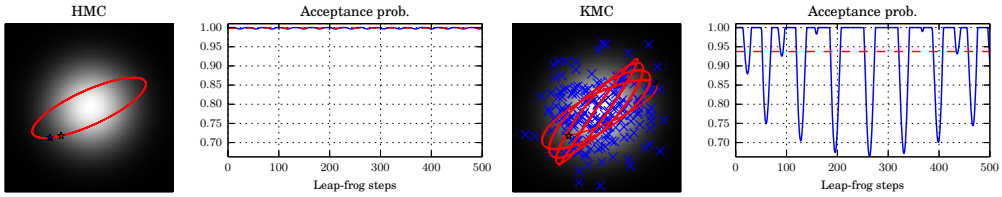


Figure 1: Hamiltonian trajectories on a 2-dimensional standard Gaussian. End points of such trajectories (red stars to blue stars) form the proposal of HMC-like algorithms. **Left:** Plain Hamiltonian trajectories oscillate on a stable orbit, and acceptance probability is close to one. **Right:** Kernel induced trajectories and acceptance probabilities on an estimated energy function.

$\pi(q)$ within $H$ with an unbiased estimator $\hat{\pi}(q)$. Note that this also involves 'recycling' the estimates of $H$ from previous iterations to ensure anyymptotic correctness, c.f. [3, Table 1]. Any deviations of the kernel induced flow from the true flow result in a decreased acceptance probability (3). We therefore need to control the approximation quality of the kernel induced potential energy to maintain high acceptance probability in practice. See Figure 1 for an illustrative example.

# 4 Two Efficient Estimators for Exponential Families in RKHS

We now address estimating the infinite dimensional exponential family model (2) from data. The original estimator in [13] has a large computational cost. This is problematic in the adaptive MCMC context, where the model has to be updated on a regular basis. We propose two efficient approximations, each with its strengths and weaknesses. Both are based on score matching.

## 4.1 Score Matching

Following [14], we model an unnormalised log probability density $\log \pi(x)$ with a parametric model

$$\log \tilde{\pi}_Z(x; f) := \log \tilde{\pi}(x; f) - \log Z(f), \tag{4}$$

where $f$ is a collection of parameters of yet unspecified dimension (c.f. natural parameters of (2)), and $Z(f)$ is an unknown normalising constant. We aim to find $\hat{f}$ from a set of $n$ samples[4] $\mathcal{D} := \{x_i\}_{i=1}^n \sim \pi$ such that $\pi(x) \approx \tilde{\pi}(x; \hat{f}) \times \text{const}$. From [14, Eq. 2], the criterion being optimised is the expected squared distance between gradients of the log density, so-called *score functions*,

$$J(f) = \frac{1}{2} \int_{\mathcal{X}} \pi(x) \left\| \nabla \log \tilde{\pi}(x; f) - \nabla \log \pi(x) \right\|_2^2 dx,$$

where we note that the normalising constants vanish from taking the gradient $\nabla$. As shown in [14, Theorem 1], it is possible to compute an empirical version *without* accessing $\pi(x)$ or $\nabla \log \pi(x)$ other than through observed samples,

$$\hat{J}(f) = \frac{1}{n} \sum_{x \in \mathcal{D}} \sum_{\ell=1}^{d} \left[ \frac{\partial^2 \log \tilde{\pi}(x; f)}{\partial x_\ell^2} + \frac{1}{2} \left( \frac{\partial \log \tilde{\pi}(x; f)}{\partial x_\ell} \right)^2 \right]. \tag{5}$$

Our approximations of the original model (2) are based on minimising (5) using approximate scores.

## 4.2 Infinite Dimensional Exponential Families Lite

The original estimator of $f$ in (2) takes a dual form in a RKHS sub-space spanned by $nd + 1$ kernel derivatives, [13, Thm. 4]. The update of the proposal at the iteration $t$ of MCMC requires inversion of a $(td + 1) \times (td + 1)$ matrix. This is clearly prohibitive if we are to run even a moderate number of iterations of a Markov chain. Following [12], we take a simple approach to avoid prohibitive computational costs in $t$: we form a proposal using a random sub-sample of fixed size $n$ from the Markov chain history, $\mathbf{z} := \{z_i\}_{i=1}^n \subseteq \{x_i\}_{i=1}^t$. In order to avoid excessive computation when $d$ is large, we replace the full dual solution with a solution in terms of span $(\{k(z_i, \cdot)\}_{i=1}^n)$, which covers the support of the true density by construction, and grows with increasing $n$. That is, we assume that the model (4) takes the 'light' form

[4] We assume a fixed sample set here but will use both the full chain history $\{x_i\}_{i=1}^t$ or a sub-sample later.

$$f(x) = \sum_{i=1}^{n} \alpha_i k(z_i, x), \tag{6}$$

where $\alpha \in \mathbb{R}^n$ are real valued parameters that are obtained by minimising the empirical score matching objective (5). This representation is of a form similar to [20, Section 4.1], the main differences being that the basis functions are chosen randomly, the basis set grows with $n$, and we will require an additional regularising term. The estimator is summarised in the following proposition, which is proved in Appendix A.

**Proposition 1.** *Given a set of samples $\mathbf{z} = \{z_i\}_{i=1}^n$ and assuming $f(x) = \sum_{i=1}^n \alpha_i k(z_i, x)$ for the Gaussian kernel of the form $k(x, y) = \exp\left(-\sigma^{-1}\|x - y\|_2^2\right)$, and $\lambda > 0$, the unique minimiser of the $\lambda\|f\|_{\mathcal{H}}^2$-regularised empirical score matching objective* (5) *is given by*

$$\hat{\alpha}_\lambda = -\frac{\sigma}{2}(C + \lambda I)^{-1} b, \tag{7}$$

*where $b \in \mathbb{R}^n$ and $C \in \mathbb{R}^{n \times n}$ are given by*

$$b = \sum_{\ell=1}^{d} \left( \frac{2}{\sigma}(K s_\ell + D_{s_\ell} K \mathbf{1} - 2 D_{x_\ell} K x_\ell) - K\mathbf{1} \right) \text{ and } C = \sum_{\ell=1}^{d} \left[ D_{x_\ell} K - K D_{x_\ell} \right] \left[ K D_{x_\ell} - D_{x_\ell} K \right],$$

*with entry-wise products $s_\ell := x_\ell \odot x_\ell$ and $D_x := \mathrm{diag}(x)$.*

The estimator costs $\mathcal{O}(n^3 + dn^2)$ computation (for computing $C, b$, and for inverting $C$) and $\mathcal{O}(n^2)$ storage, for a fixed random chain history sub-sample size $n$. This can be further reduced via low-rank approximations to the kernel matrix and conjugate gradient methods, which are derived in Appendix A.

Gradients of the model are given as $\nabla f(x) = \sum_{i=1}^n \alpha_i \nabla k(x, x_i)$, i.e. they simply require to evaluate gradients of the kernel function. Evaluation and storage of $\nabla f(\cdot)$ both cost $\mathcal{O}(dn)$.

### 4.3 Exponential Families in Finite Feature Spaces

Instead of fitting an infinite-dimensional model on a subset of the available data, the second estimator is based on fitting a finite dimensional approximation using *all* available data $\{x_i\}_{i=1}^t$, in *primal* form. As we will see, updating the estimator when a new data point arrives can be done online.

Define an $m$-dimensional approximate feature space $\mathcal{H}_m = \mathbb{R}^m$, and denote by $\phi_x \in \mathcal{H}_m$ the embedding of a point $x \in \mathcal{X} = \mathbb{R}^d$ into $\mathcal{H}_m = \mathbb{R}^m$. Assume that the embedding approximates the kernel function as a finite rank expansion $k(x, y) \approx \phi_x^\top \phi_y$. The log unnormalised density of the infinite model (2) can be approximated by assuming the model in (4) takes the form

$$f(x) = \langle \theta, \phi_x \rangle_{\mathcal{H}_m} = \theta^\top \phi_x \tag{8}$$

To fit $\theta \in \mathbb{R}^m$, we again minimise the score matching objective (5), as proved in Appendix B.

**Proposition 2.** *Given a set of samples $\mathbf{x} = \{x_i\}_{i=1}^t$ and assuming $f(x) = \theta^\top \phi_x$ for a finite dimensional feature embedding $x \mapsto \phi_x \in \mathbb{R}^m$, and $\lambda > 0$, the unique minimiser of the $\lambda\|\theta\|_2^2$-regularised empirical score matching objective* (5) *is given by*

$$\hat{\theta}_\lambda := (C + \lambda I)^{-1} b, \tag{9}$$

*where*

$$b := -\frac{1}{n} \sum_{i=1}^{t} \sum_{\ell=1}^{d} \ddot{\phi}_{x_i}^\ell \in \mathbb{R}^m, \qquad C := \frac{1}{n} \sum_{i=1}^{t} \sum_{\ell=1}^{d} \dot{\phi}_{x_i}^\ell \left( \dot{\phi}_{x_i}^\ell \right)^T \in \mathbb{R}^{m \times m},$$

*with $\dot{\phi}_x^\ell := \frac{\partial}{\partial x_\ell} \phi_x$ and $\ddot{\phi}_x^\ell := \frac{\partial^2}{\partial x_\ell^2} \phi_x$.*

An example feature embedding based on random Fourier features [17, 21] and a standard Gaussian kernel is $\phi_x = \sqrt{\frac{2}{m}} \left[ \cos(\omega_1^T x + u_1), \ldots, \cos(\omega_m^T x + u_m) \right]$, with $\omega_i \sim \mathcal{N}(\omega)$ and $u_i \sim \texttt{Uniform}[0, 2\pi]$. The estimator has a one-off cost of $\mathcal{O}(tdm^2 + m^3)$ computation and $\mathcal{O}(m^2)$ storage. Given that we have computed a solution based on the Markov chain history $\{x_i\}_{i=1}^t$, however, it is straightforward to update $C, b$, and the solution $\hat{\theta}_\lambda$ online, after a new point $x_{t+1}$ arrives. This is achieved by storing running averages and performing low-rank updates of matrix inversions, and costs $\mathcal{O}(dm^2)$ computation and $\mathcal{O}(m^2)$ storage, *independent* of $t$. Further details are given in Appendix B.

Gradients of the model are $\nabla f(x) = \left[ \nabla \phi_x \right]^\top \hat{\theta}$ , i.e., they require the evaluation of the gradient of the feature space embedding, costing $\mathcal{O}(md)$ computation and and $\mathcal{O}(m)$ storage.

**Algorithm 1 Kernel Hamiltonian Monte Carlo – Pseudo-code**

***Input*:** Target (possibly noisy estimator) $\hat{\pi}$, adaptation schedule $a_t$, HMC parameters, Size of basis $m$ or sub-sample size $n$.
At iteration $t+1$, current state $x_t$, history $\{x_i\}_{i=1}^t$, perform (1-4) with probability $a_t$

**KMC lite:**

1. Update sub-sample $\mathbf{z} \subseteq \{x_i\}_{i=1}^t$
2. Re-compute $C, b$ from Prop. 1
3. Solve $\hat{\alpha}_\lambda = -\frac{\sigma}{2}(C + \lambda I)^{-1} b$
4. $\nabla f(x) \leftarrow \sum_{i=1}^n \alpha_i \nabla k(x, z_i)$

**KMC finite:**

1. Update to $C, b$ from Prop. 2
2. Perform rank-$d$ update to $C^{-1}$
3. Update $\hat{\theta}_\lambda = (C + \lambda I)^{-1} b$
4. $\nabla f(x) \leftarrow [\nabla \phi_x]^\top \hat{\theta}$

5. Propose $(p', x^*)$ with kernel induced Hamiltonian flow, using $\nabla_x U = \nabla_x f$
6. Perform Metropolis step using $\hat{\pi}$: accept $x_{t+1} \leftarrow x^*$ w.p. (3) and reject $x_{t+1} \leftarrow x_t$ otherwise
   If $\hat{\pi}$ is noisy and $x^*$ was accepted, store above $\hat{\pi}(x^*)$ for evaluating (3) in the next iteration

## 5 Kernel Hamiltonian Monte Carlo

Constructing a kernel induced Hamiltonian flow as in Section 3 from the gradients of the infinite dimensional exponential family model (2), and approximate estimators (6),(8), we arrive at a gradient free, adaptive MCMC algorithm: *Kernel Hamiltonian Monte Carlo* (Algorithm 1).

**Computational Efficiency, Geometric Ergodicity, and Burn-in** KMC finite using (8) allows for online updates using the *full* Markov chain history, and therefore is a more elegant solution than KMC lite, which has greater computational cost and requires sub-sampling the chain history. Due to the parametric nature of KMC finite, however, the tails of the estimator are not guaranteed to decay. For example, the random Fourier feature embedding described below Proposition 2 contains periodic cosine functions, and therefore oscillates in the tails of (8), resulting in a reduced acceptance probability. As we will demonstrate in the experiments, this problem does not appear when KMC finite is initialised in high-density regions, nor after burn-in. In situations where information about the target density support is unknown, and during burn-in, we suggest to use the lite estimator (7), whose gradients decay outside of the training data. As a result, KMC lite is guaranteed to fall back to a Random Walk Metropolis in unexplored regions, inheriting its convergence properties, and smoothly transitions to HMC-like proposals as the MCMC chain grows. A proof of the proposition below can be found in Appendix C.

**Proposition 3.** *Assume $d = 1$, $\pi(x)$ has log-concave tails, the regularity conditions of [22, Thm 2.2] (implying $\pi$-irreducibility and smallness of compact sets), that MCMC adaptation stops after a fixed time, and a fixed number $L$ of $\epsilon$-leapfrog steps. If $\limsup_{\|x\|_2 \to \infty} \|\nabla f(x)\|_2 = 0$, and $\exists M : \forall x : \|\nabla f(x)\|_2 \leq M$, then KMC lite is geometrically ergodic from $\pi$-almost any starting point.*

**Vanishing adaptation** MCMC algorithms that use the history of the Markov chain for constructing proposals might not be asymptotically correct. We follow [12, Sec. 4.2] and the idea of 'vanishing adaptation' [11], to avoid such biases. Let $\{a_t\}_{i=0}^\infty$ be a schedule of decaying probabilities such that $\lim_{t\to\infty} a_t = 0$ and $\sum_{t=0}^\infty a_t = \infty$. We update the density gradient estimate according to this schedule in Algorithm 1. Intuitively, adaptation becomes less likely as the MCMC chain progresses, but never fully stops, while sharing asymptotic convergence with adaptation that stops at a fixed point [23, Theorem 1]. Note that Proposition 3 is a stronger statement about the *convergence rate*.

**Free Parameters** KMC has two free parameters: the Gaussian kernel bandwidth $\sigma$, and the regularisation parameter $\lambda$. As KMC's performance depends on the quality of the approximate infinite dimensional exponential family model in (6) or (8), a principled approach is to use the score matching objective function in (5) to choose $\sigma, \lambda$ pairs via cross-validation (using e.g. 'hot-started' black-box optimisation). Earlier adaptive kernel-based MCMC methods [12] did not address parameter choice.

## 6 Experiments

We start by quantifying performance of KMC finite on synthetic targets. We emphasise that these results can be reproduced with the lite version.

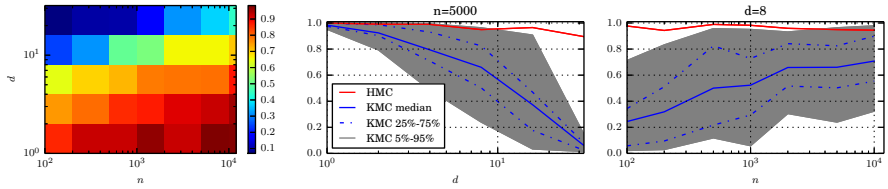


Figure 2: Hypothetical acceptance probability of KMC finite on a challening target in growing dimensions. **Left:** As a function of $n = m$ (x-axis) and $d$ (y-axis). **Middle/right:** Slices through left plot with error bars for fixed $n = m$ and as a function of $d$ (left), and for fixed $d$ as a function of $n = m$ (right).

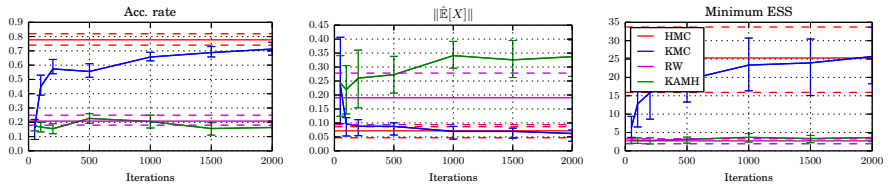


Figure 3: Results for the 8-dimensional synthetic Banana. As the amout of observed data increases, KMC performance approaches HMC – outperforming KAMH and RW. 80% error bars over 30 runs.

**KMC Finite: Stability of Trajectories in High Dimensions** In order to quantify efficiency in growing dimensions, we study hypothetical acceptance rates along trajectories on the kernel induced Hamiltonian flow (no MCMC yet) on a challenging Gaussian target: We sample the diagonal entries of the covariance matrix from a `Gamma(1,1)` distribution and rotate with a uniformly sampled random orthogonal matrix. The resulting target is challenging to estimate due to its 'non-singular smoothness', i.e., substantially differing length-scales across its principal components. As a single Gaussian kernel is not able to effeciently represent such scaling families, we use a rational quadratic kernel for the gradient estimation, whose random features are straightforward to compute. Figure 2 shows the average acceptance over 100 independent trials as a function of the number of (ground truth) samples and basis functions, which are set to be equal $n = m$, and of dimension $d$. In low to moderate dimensions, gradients of the finite estimator lead to acceptance rates comparable to plain HMC. On targets with more 'regular' smoothness, the estimator performs well in up to $d \approx 100$, with less variance. See Appendix D.1 for details.

**KMC Finite: HMC-like Mixing on a Synthetic Example** We next show that KMC's performance approaches that of HMC as it sees more data. We compare KMC, HMC, an isotropic random walk (RW), and KAMH on the 8-dimensional nonlinear banana-shaped target; see Appendix D.2. We here only quantify mixing *after* a sufficient burn-in (burn-in speed is included in next example). We quantify performance on estimating the target's mean, which is exactly $\mathbf{0}$. We tuned the scaling of KAMH and RW to achieve 23% acceptance. We set HMC parameters to achieve 80% acceptance and then used the same parameters for KMC. We ran all samplers for 2000+200 iterations from a random start point, discarded the burn-in and computed acceptance rates, the norm of the empirical mean $\|\hat{\mathbb{E}}[x]\|$, and the minimum effective sample size (ESS) across dimensions. For KAMH and KMC, we repeated the experiment for an increasing number of burn-in samples and basis functions $m = n$. Figure 3 shows the results as a function of $m = n$. KMC clearly outperforms RW and KAMH, and eventually achieves performance close to HMC as $n = m$ grows.

**KMC Lite: Pseudo-Marginal MCMC for GP Classification on Real World Data** We next apply KMC to sample from the marginal posterior over hyper-parameters of a Gaussian Process Classification (GPC) model on the UCI Glass dataset [24]. Classical HMC cannot be used for this problem, due to the intractability of the marginal data likelihood. Our experimental protocol mostly follows [12, Section 5.1], see Appendix D.3, but uses only 6000 MCMC iterations *without* discarding a burn-in, i.e., we study how fast KMC initially explores the target. We compare convergence in terms of all mixed moments of order up to 3 to a set of benchmark samples (MMD [25], lower is better). KMC randomly uses between 1 and 10 leapfrog steps of a size chosen uniformly in $[0.01, 0.1]$,

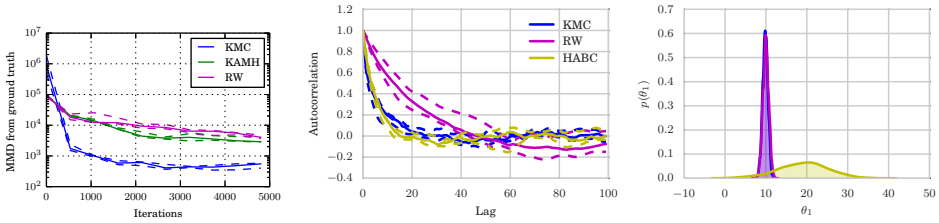


Figure 4: **Left:** Results for 9-dimensional marginal posterior over length scales of a GPC model applied to the UCI Glass dataset. The plots shows convergence (no burn-in discarded) of all mixed moments up to order 3 (lower MMD is better). **Middle/right:** ABC-MCMC auto-correlation and marginal $\theta_1$ posterior for a 10-dimensional skew normal likelihood. While KMC mixes as well as HABC, it does not suffer from any bias (overlaps with RW, while HABC is significantly different) and requires fewer simulations per proposal.

a standard Gaussian momentum, and a kernel tuned by cross-validation, see Appendix D.3. We did not extensively tune the HMC parameters of KMC as the described settings were sufficient. Both KMC and KAMH used 1000 samples from the chain history. Figure 4 (left) shows that KMC's burn-in contains a short 'exploration phase' where produced estimates are bad, due to it falling back to a random walk in unexplored regions, c.f. Proposition 3. From around 500 iterations, however, KMC clearly outperforms both RW and the earlier state-of-the-art KAMH. These results are backed by the minimum ESS (not plotted), which is around 415 for KMC and is around 35 and 25 for KAMH and RW, respectively. Note that all samplers effectively stop improving from 3000 iterations – indicating a burn-in bias. All samplers took 1h time, with most time spent estimating the marginal likelihood.

**KMC Lite: Reduced Simulations and no Additional Bias in ABC** We now apply KMC in the context of Approximate Bayesian Computation (ABC), which often is employed when the data likelihood is intractable but can be obtained by simulation, see e.g. [6]. ABC-MCMC [5] targets an approximate posterior by constructing an unbiased Monte Carlo estimator of the approximate likelihood. As each such evaluation requires expensive simulations from the likelihood, the goal of all ABC methods is to reduce the number of such simulations. Accordingly, Hamiltonian ABC was recently proposed [8], combining the synthetic likelihood approach [26] with gradients based on stochastic finite differences. We remark that this requires to simulate from the likelihood in *every* leapfrog step, and that the additional bias from the Gaussian likelihood approximation can be problematic. In contrast, KMC does not require simulations to construct a proposal, but rather 'invests' simulations into an accept/reject step (3) that ensures convergence to the *original* ABC target. Figure 4 (right) compares performance of RW, HABC (sticky random numbers and SPAS, [8, Sec. 4.3, 4.4]), and KMC on a 10-dimensional skew-normal distribution $p(y|\theta) = 2\mathcal{N}(\theta, I)\,\Phi(\langle \alpha, y \rangle)$ with $\theta = \alpha = \mathbf{1} \cdot 10$. KMC mixes as well as HABC, but HABC suffers from a severe bias. KMC also reduces the number of simulations per proposal by a factor $2L = 100$. See Appendix D.4 for details.

## 7 Discussion

We have introduced KMC, a kernel-based gradient free adaptive MCMC algorithm that mimics HMC's behaviour by estimating target gradients in an RKHS. In experiments, KMC outperforms random walk based sampling methods in up to $d = 50$ dimensions, including the recent kernel-based KAMH [12]. KMC is particularly useful when gradients of the target density are unavailable, as in PM-MCMC or ABC-MCMC, where classical HMC cannot be used. We have proposed two efficient empirical estimators for the target gradients, each with different strengths and weaknesses, and have given experimental evidence for the robustness of both.

Future work includes establishing theoretical consistency and uniform convergence rates for the empirical estimators, for example via using recent analysis of random Fourier Features with tight bounds [21], and a thorough experimental study in the ABC-MCMC context where we see a lot of potential for KMC. It might also be possible to use KMC as a precomputing strategy to speed up classical HMC as in [27]. For code, see `https://github.com/karlnapf/kernel_hmc`

## References


[1] R.M. Neal. MCMC using Hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*, 2, 2011.

[2] M.A. Beaumont. Estimation of population growth or decline in genetically monitored populations. *Genetics*, 164(3):1139–1160, 2003.

[3] C. Andrieu and G.O. Roberts. The pseudo-marginal approach for efficient Monte Carlo computations. *The Annals of Statistics*, 37(2):697–725, April 2009.

[4] M. Filippone and M. Girolami. Pseudo-marginal Bayesian inference for Gaussian Processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2014.

[5] P. Marjoram, J. Molitor, V. Plagnol, and S. Tavaré. Markov chain Monte Carlo without likelihoods. *Proceedings of the National Academy of Sciences*, 100(26):15324–15328, 2003.

[6] S.A. Sisson and Y. Fan. Likelihood-free Markov chain Monte Carlo. *Handbook of Markov chain Monte Carlo*, 2010.

[7] T. Chen, E. Fox, and C. Guestrin. Stochastic Gradient Hamiltonian Monte Carlo. In *ICML*, pages 1683–1691, 2014.

[8] E. Meeds, R. Leenders, and M. Welling. Hamiltonian ABC. In *UAI*, 2015.

[9] M. Betancourt. The Fundamental Incompatibility of Hamiltonian Monte Carlo and Data Subsampling. *arXiv preprint arXiv:1502.01510*, 2015.

[10] H. Haario, E. Saksman, and J. Tamminen. Adaptive proposal distribution for random walk Metropolis algorithm. *Computational Statistics*, 14(3):375–395, 1999.

[11] C. Andrieu and J. Thoms. A tutorial on adaptive MCMC. *Statistics and Computing*, 18(4):343–373, December 2008.

[12] D. Sejdinovic, H. Strathmann, M. Lomeli, C. Andrieu, and A. Gretton. Kernel Adaptive Metropolis-Hastings. In *ICML*, 2014.

[13] B. Sriperumbudur, K. Fukumizu, R. Kumar, A. Gretton, and A. Hyvärinen. Density Estimation in Infinite Dimensional Exponential Families. *arXiv preprint arXiv:1312.3516*, 2014.

[14] A. Hyvärinen. Estimation of non-normalized statistical models by score matching. *JMLR*, 6:695–709, 2005.

[15] Larry Wasserman. *All of nonparametric statistics*. Springer, 2006.

[16] C.E. Rasmussen. Gaussian Processes to Speed up Hybrid Monte Carlo for Expensive Bayesian Integrals. *Bayesian Statistics 7*, pages 651–659, 2003.

[17] A. Rahimi and B. Recht. Random features for large-scale kernel machines. In *NIPS*, pages 1177–1184, 2007.

[18] M. Betancourt, S. Byrne, and M. Girolami. Optimizing The Integrator Step Size for Hamiltonian Monte Carlo. *arXiv preprint arXiv:1503.01916*, 2015.

[19] A. Berlinet and C. Thomas-Agnan. *Reproducing Kernel Hilbert Spaces in Probability and Statistics*. Kluwer, 2004.

[20] A. Hyvärinen. Some extensions of score matching. *Computational Statistics & Data Analysis*, 51:2499–2512, 2007.

[21] B.K. Sriperumbudur and Z. Szabó. Optimal rates for random Fourier features. In *NIPS*, 2015.

[22] G.O. Roberts and R.L. Tweedie. Geometric convergence and central limit theorems for multidimensional Hastings and Metropolis algorithms. *Biometrika*, 83(1):95–110, 1996.

[23] G.O. Roberts and J.S. Rosenthal. Coupling and ergodicity of adaptive Markov chain Monte Carlo algorithms. *Journal of Applied Probability*, 44(2):458–475, 03 2007.

[24] K. Bache and M. Lichman. UCI Machine Learning Repository, 2013.

[25] A. Gretton, K. Borgwardt, B. Schölkopf, A. J. Smola, and M. Rasch. A kernel two-sample test. *JMLR*, 13:723–773, 2012.

[26] S. N. Wood. Statistical inference for noisy nonlinear ecological dynamic systems. *Nature*, 466(7310):1102–1104, 08 2010.

[27] C. Zhang, B. Shahbaba, and H. Zhao. Hamiltonian Monte Carlo Acceleration Using Neural Network Surrogate functions. *arXiv preprint arXiv:1506.05555*, 2015.

[28] J. Shawe-Taylor and N. Cristianini. *Kernel methods for pattern analysis*. Cambridge university press, 2004.

[29] Q. Le, T. Sarlós, and A. Smola. Fastfood–approximating kernel expansions in loglinear time. In *ICML*, 2013.

[30] K.L. Mengersen and R.L. Tweedie. Rates of convergence of the Hastings and Metropolis algorithms. *The Annals of Statistics*, 24(1):101–121, 1996.

# Appendix

The Appendix contains proofs for Propositions 1 and 2, as well as additional computational details for both KMC lite in Section A and KMC finite in Section B. Section C covers the proof of geometric ergodicity of KMC lite from Proposition 3. Section D describes further experimental details.

## A Lite Estimator

**Proof of Proposition 1**

The proof below extends the model in [20, Section 4.1]. We assume that the model log-density (4) takes the form in Proposition 1, then directly implement score functions (5), from which we derive an empirical score matching objective as a system of linear equations.

*Proof.* As assumed the log unnormalised density takes the form

$$f(x) = \sum_{i=1}^{n} \alpha_i k(x_i, x)$$

where $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is the Gaussian kernel in the form

$$k(x_i, x) = \exp\left(-\frac{1}{\sigma}\|x_i - x\|^2\right) = \exp\left(-\frac{1}{\sigma}\sum_{\ell=1}^{d}(x_{i\ell} - x_\ell)^2\right).$$

The score functions for (5) are then given by

$$\psi_\ell(x;\alpha) := \frac{\partial \log \tilde{\pi}(x;f)}{\partial x_\ell} = \frac{2}{\sigma}\sum_{i=1}^{n}\alpha_i(x_{i\ell} - x_\ell)\exp\left(-\frac{\|x_i - x\|^2}{\sigma}\right),$$

and

$$\begin{aligned}
\partial_\ell\psi_\ell(x;\alpha) :=& \frac{\partial^2 \log \tilde{\pi}(x;f)}{\partial^2 x_\ell} \\
&= -\frac{2}{\sigma}\sum_{i=1}^{n}\alpha_i \exp\left(-\frac{\|x_i - x\|^2}{\sigma}\right) + \left(\frac{2}{\sigma}\right)^2 \sum_{i=1}^{n}\alpha_i(x_{i\ell} - x_\ell)^2 \exp\left(-\frac{\|x_i - x\|^2}{\sigma}\right) \\
&= \frac{2}{\sigma}\sum_{i=1}^{n}\alpha_i \exp\left(-\frac{\|x_i - x\|^2}{\sigma}\right)\left[-1 + \frac{2}{\sigma}(x_{i\ell} - x_\ell)^2\right].
\end{aligned}$$

Substituting this into (5) yields

$$\begin{aligned}
J(\alpha) &= \frac{1}{n}\sum_{i=1}^{n}\sum_{\ell=1}^{d}\left[\partial_\ell\psi_\ell(x_i;\alpha) + \frac{1}{2}\psi_\ell(x_i;\alpha)^2\right] \\
&= \frac{2}{n\sigma}\sum_{\ell=1}^{d}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i \exp\left(-\frac{\|x_i - x_j\|^2}{\sigma}\right)\left[-1 + \frac{2}{\sigma}(x_{i\ell} - x_{j\ell})^2\right] \\
&\quad + \frac{2}{n\sigma^2}\sum_{\ell=1}^{d}\sum_{i=1}^{n}\left[\sum_{j=1}^{n}\alpha_j(x_{j\ell} - x_{i\ell})\exp\left(-\frac{\|x_i - x_j\|^2}{\sigma}\right)\right]^2.
\end{aligned}$$

We now rewrite $J(\alpha)$ in matrix form. The expression for the term $J(\alpha)$ being optimised is the sum of two terms.

**First Term**:

$$\sum_{\ell=1}^{d}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i \exp\left(-\frac{\|x_i - x_j\|^2}{\sigma}\right)\left[-1 + \frac{2}{\sigma}(x_{i\ell} - x_{j\ell})^2\right]$$

We only need to compute

$$\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i \exp\left(-\frac{\|x_i - x_j\|^2}{\sigma}\right)(x_{i\ell} - x_{j\ell})^2$$
$$=\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i \exp\left(-\frac{\|x_i - x_j\|^2}{\sigma}\right)\left(x_{i\ell}^2 + x_{j\ell}^2 - 2x_{i\ell}x_{j\ell}\right).$$

Define

$$x_\ell := \begin{bmatrix} x_{1\ell} & \dots & x_{m\ell} \end{bmatrix}^\top.$$

The final term may be computed with the right ordering of operations,

$$-2(\alpha \odot x_\ell)^\top K x_\ell,$$

where $\alpha \odot x_\ell$ is the entry-wise product. The remaining terms are sums with constant row or column terms. Define $s_\ell := x_\ell \odot x_\ell$ with components $s_{i\ell} = x_{i\ell}^2$. Then

$$\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i k_{ij} s_{j\ell} = \alpha^\top K s_\ell.$$

Likewise

$$\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i x_{i\ell}^2 k_{ij} = (\alpha \odot s_\ell)^\top K\mathbf{1}.$$

**Second Term**: Considering only the $\ell$-th dimension, this is

$$\sum_{i=1}^{n}\left[\sum_{j=1}^{n}\alpha_j(x_{j\ell} - x_{i\ell})\exp\left(-\frac{\|x_i - x_j\|^2}{\sigma}\right)\right]^2.$$

In matrix notation, the inner sum is a column vector,

$$K(\alpha \odot x_\ell) - (K\alpha) \odot x_\ell.$$

We take the entry-wise square and sum the resulting vector. Denote by $D_x := \operatorname{diag}(x)$, then the following two relations hold

$$K(\alpha \odot x) = KD_x\alpha,$$
$$(K\alpha) \odot x = D_x K\alpha.$$

This means that $J(\alpha)$ as defined previously,

$$J(\alpha) = \frac{2}{n\sigma}\sum_{\ell=1}^{d}\left[\frac{2}{\sigma}\left[\alpha^T K s_\ell + (\alpha \odot s_\ell)^\top K\mathbf{1} - 2(\alpha \odot x_\ell)^\top K x_\ell\right] - \alpha^T K\mathbf{1}\right]$$
$$+ \frac{2}{n\sigma^2}\sum_{\ell=1}^{d}\left[(\alpha \odot x_\ell)^\top K - x_\ell^\top \odot (\alpha^\top K)\right]\left[K(\alpha \odot x_\ell) - (K\alpha) \odot x_\ell\right],$$

can be rewritten as

$$J(\alpha) = \frac{2}{n\sigma}\alpha^T\sum_{\ell=1}^{d}\left[\frac{2}{\sigma}(Ks_\ell + D_{s_\ell}K\mathbf{1} - 2D_{x_\ell}Kx_\ell) - K\mathbf{1}\right]$$
$$+ \frac{2}{n\sigma^2}\alpha^T\left(\sum_{\ell=1}^{d}\left[D_{x_\ell}K - KD_{x_\ell}\right]\left[KD_{x_\ell} - D_{x_\ell}K\right]\right)\alpha$$
$$= \frac{2}{n\sigma}\alpha^T b + \frac{2}{n\sigma^2}\alpha^\top C\alpha,$$

where

$$b = \sum_{\ell=1}^{d} \left( \frac{2}{\sigma}(Ks_\ell + D_{s_\ell} K\mathbf{1} - 2D_{x_\ell} Kx_\ell) - K\mathbf{1} \right) \in \mathbb{R}^n,$$

$$C = \sum_{\ell=1}^{d} \left[ D_{x_\ell} K - KD_{x_\ell} \right] \left[ KD_{x_\ell} - D_{x_\ell} K \right] \in \mathbb{R}^{n \times n}.$$

Assuming $C$ is invertible, this is minimised by

$$\hat{\alpha} = -\frac{\sigma}{2} C^{-1} b.$$

□

As in [13], we add a term $\lambda \|f\|^2_{\mathcal{H}}$ for $\lambda \in \mathbb{R}^+$, in order to control the norm of the natural parameters in the RKHS $\|f\|^2_{\mathcal{H}}$. This results in the regularised and numerically more stable solution $\hat{\alpha}_\lambda := (C + \lambda I)^{-1} b$.

**Reduced Computational Costs via Low-rank Approximations and Conjugate Gradient**

Solving the linear system in (7) requires $\mathcal{O}(n^3)$ computation and $\mathcal{O}(n^2)$ storage for a fixed random sub-sample of the chain history $\mathbf{z}$. In order to allow for large $n$, and to exploit potential manifold structure in the RKHS, we apply a low-rank approximation to the kernel matrix via incomplete Cholesky [28, Alg. 5.12], that is a standard way to achieve linear computational costs for kernel methods. We rewrite the kernel matrix

$$K \approx LL^\top,$$

where $L \in \mathbb{R}^{n \times \ell}$ is obtained via dual partial Gram–Schmidt orthonormalisation and costs both $\mathcal{O}(n\ell)$ computation and storage. Usually $\ell \ll n$, and $\ell$ can be chosen via an accuracy cut-off parameter on the kernel spectrum in the same fashion as for other low-rank approximations, such as PCA[5]. Given such a representation of $K$, we can rewrite any matrix-vector product as

$$Kb \approx (LL^\top)b = L(L^\top b),$$

where each left multiplication of $L$ costs $\mathcal{O}(n\ell)$ and we never need to store $LL^\top$. This idea can be used to achieve costs of $\mathcal{O}(n\ell)$ when computing $b$, and left-multiplying $C$. Combining the technique with conjugate gradient (CG) allows to solve (7) with a maximum of $n$ such matrix-vector products, yielding a total computational cost of $\mathcal{O}(n^2\ell)$. In practice, we can monitor residuals and stop CG after a fixed number of iterations $\tau \ll n$, where $\tau$ depends on the decay of the spectrum of $K$. We arrive at a total cost of $\mathcal{O}(n\ell\tau)$ computation and $\mathcal{O}(n\ell)$ storage. CG also has the advantage of allowing for 'hot starts', i.e. initialising the linear solver at a previous solution. Further details can be found in our implementation.

## B Finite Feature Space Estimator

**Proof of Proposition 2**

We assume the model log-density (4) takes the primal form in a finite dimensional feature space as in Proposition 2, then again directly implement score functions in (5) and minimise it via a linear solve.

*Proof.* As assumed the log unnormalised density takes the form

$$f(x) = \langle \theta, \phi_x \rangle_{\mathcal{H}_m} = \theta^\top \phi_x,$$

[5] In this paper, we solely use the Gaussian kernel, whose spectrum decays exponentially fast.

where $x \in \mathbb{R}^d$ is embedded into a finite dimensional feature space $\mathcal{H}_m = \mathbb{R}^m$ as $x \mapsto \phi_x$. The score functions in (5) then can be written as the simple linear form

$$\psi_\ell(\xi;\theta) := \frac{\partial \log \tilde{\pi}(x;\theta)}{\partial x_\ell} = \theta^\top \dot{\phi}_x^\ell \quad \text{and} \quad \partial_\ell \psi_\ell(\xi;\theta) := \frac{\partial^2 \log \tilde{\pi}(x;\theta)}{\partial x_\ell^2} = \theta^\top \ddot{\phi}_x^\ell, \tag{10}$$

where we defined the $m$-dimensional feature vector derivatives $\dot{\phi}_x^\ell := \frac{\partial}{\partial x_\ell}\phi_x$ and $\ddot{\phi}_x^\ell := \frac{\partial^2}{\partial x_\ell^2}\phi_x$. Plugging those into the empirical score matching objective in (5), we arrive at

$$\begin{aligned} J(\theta) &= \frac{1}{n}\sum_{i=1}^n \sum_{\ell=1}^d \left[\partial_\ell \psi_\ell(x_i;\theta) + \frac{1}{2}\psi_\ell^2(x_i;\theta)\right] \\ &= \frac{1}{n}\sum_{i=1}^n \sum_{\ell=1}^d \left[\theta^T \ddot{\phi}_{x_i}^\ell + \frac{1}{2}\theta^T \left(\dot{\phi}_{x_i}^\ell \left(\dot{\phi}_{x_i}^\ell\right)^T\right)\theta\right] \\ &= \frac{1}{2}\theta^T C \theta - \theta^T b \end{aligned} \tag{11}$$

where

$$b := -\frac{1}{n}\sum_{i=1}^n \sum_{\ell=1}^d \ddot{\phi}_{x_i}^\ell \in \mathbb{R}^m \quad \text{and} \quad C := \frac{1}{n}\sum_{i=1}^n \sum_{\ell=1}^d \left(\dot{\phi}_{x_i}^\ell \left(\dot{\phi}_{x_i}^\ell\right)^\top\right) \in \mathbb{R}^{m\times m}. \tag{12}$$

Assuming that $C$ is invertible (trivial for $n \geq m$), the objective is uniquely minimised by differentiating (11) wrt. $\theta$, setting to zero, and solving for $\theta$. This gives

$$\hat{\theta} := C^{-1} b. \tag{13}$$

□

Again, similar to [13], we add a term $\lambda/2\|\theta\|_2^2$ for $\lambda \in \mathbb{R}^+$ to (11), in order to control the norm of the natural parameters $\theta \in \mathcal{H}^m$. This results in the regularised and numerically more stable solution $\hat{\theta}_\lambda := (C + \lambda I)^{-1} b$.

Next, we give an example for the approximate feature space $\mathcal{H}_m$. Note that the above approach can be combined with *any* set of finite dimensional approximate feature mappings $\phi_x$.

**Example: Random Fourier Features for the Gaussian Kernel**

We now combine the finite dimensional approximate infinite dimensional exponential family model with the "random kitchen sink" [17]. Assume a translation invariant kernel $k(x,y) = \tilde{k}(x-y)$. Bochner's theorem gives the representation

$$k(x,y) = \tilde{k}(x-y) = \int_{\mathbb{R}^d} \exp\left(i\omega^\top (x-y)\right) \mathrm{d}\Gamma(\omega),$$

where $\Gamma(\omega)$ is the Fourier transform of the kernel. An approximate feature mapping for such kernels can be obtained via dropping imaginary terms and approximating the integral with Monte Carlo integration. This gives

$$\phi_x = \sqrt{\frac{2}{m}}\left[\cos(\omega_1^\top x + u_1), \ldots, \cos(\omega_m^\top x + u_m)\right],$$

with fixed random basis vector realisations that depend on the kernel via $\Gamma(\omega)$,

$$\omega_i \sim \Gamma(\omega),$$

and fixed random offset realisations

$$u_i \sim \texttt{Uniform}[0, 2\pi],$$

for $i = 1 \ldots m$. It is easy to see that this approximation is consistent for $m \to \infty$, i.e.

$$\mathbb{E}_{\omega,b}\left[\phi_x^T \phi_y\right] = k(x,y).$$

See [17] for details and a uniform convergence bound and [21] for a more detailed analysis with tighter bounds. Note that it is possible to achieve logarithmic computational costs in $d$ exploiting properties of Hadamard matrices [29].

The feature map derivatives (10) are given by

$$\begin{aligned}\dot{\phi}^{\ell}_{\xi} &= \sqrt{\frac{2}{m}}\frac{\partial}{\partial \xi_\ell}\left[\cos(\omega_1^T\xi + u_1), \ldots, \cos(\omega_m^T\xi + u_m)\right] \\ &= -\sqrt{\frac{2}{m}}\left[\sin(\omega_1^T\xi + u_1)\omega_{1\ell}, \ldots, \sin(\omega_m^T\xi + u_m)\omega_{m\ell}\right] \\ &= -\sqrt{\frac{2}{m}}\left[\sin(\omega_1^T\xi + u_1), \ldots, \sin(\omega_m^T\xi + u_m)\right] \odot \left[\omega_{1\ell}, \ldots, \omega_{m\ell}\right],\end{aligned}$$

where $\omega_{j\ell}$ is the $\ell$-th component of $\omega_j$, and

$$\begin{aligned}\ddot{\phi}^{\ell}_{\xi} :&= -\sqrt{\frac{2}{m}}\frac{\partial}{\partial \xi_\ell}\left[\sin(\omega_1^T\xi + u_1), \ldots, \sin(\omega_m^T\xi + u_m)\right] \odot \left[\omega_{1\ell}, \ldots, \omega_{m\ell}\right] \\ &= -\sqrt{\frac{2}{m}}\left[\cos(\omega_1^T\xi + u_1), \ldots, \cos(\omega_m^T\xi + u_m)\right] \odot \left[\omega^2_{1\ell}, \ldots, \omega^2_{m\ell}\right] \\ &= -\phi_\xi \odot \left[\omega^2_{1\ell}, \ldots, \omega^2_{m\ell}\right],\end{aligned}$$

where $\odot$ is the element-wise product. Consequently the gradient is given by

$$\nabla_\xi \phi_\xi = \begin{bmatrix}\dot{\phi}^1_\xi \\ \vdots \\ \dot{\phi}^d_\xi\end{bmatrix} \in \mathbb{R}^{d\times m}.$$

As an example, the translation invariant Gaussian kernel and its Fourier transform are

$$k(x,y) = \exp\left(-\frac{\|x-y\|_2^2}{2\sigma^2}\right) \quad \text{and} \quad \Gamma(\omega) = \mathcal{N}\left(\omega\middle|\mathbf{0}, \frac{2}{\sigma^2}I_m\right).$$

**Constant Cost Updates**

A convenient property of the finite feature space approximation is that its primal representation of the solution allows to update (12) in an online fashion. When combined with MCMC, each new point $x_{t+1}$ of the Markov chain history only adds a term of the form $-\sum_{\ell=1}^d \ddot{\phi}^\ell_{x_{t+1}} \in \mathbb{R}^m$ and $\sum_{\ell=1}^d \dot{\phi}^\ell_{x_{t+1}}(\dot{\phi}^\ell_{x_{t+1}})^\top \in \mathbb{R}^{m\times m}$ to the moving averages of $b$ and $C$ respectively. Consequently, at iteration $t$, rather than fully re-computing (13) at the cost of $\mathcal{O}(tdm^2 + m^3)$ for every new point, we can use rank-$d$ updates to construct the minimiser of (11) from the solution of the previous iteration. Assume we have computed the sum of all moving average terms,

$$\bar{C}_t^{-1} := \left(\sum_{i=1}^t \sum_{\ell=1}^d \left(\dot{\phi}^\ell_{x_i}\left(\dot{\phi}^\ell_{x_i}\right)^\top\right)\right)^{-1}$$

from feature vectors derivatives $\ddot{\phi}^\ell_{x_i} \in \mathbb{R}^m$ of some set of points $\{x_i\}_{i=1}^t$, and subsequently receive receive a new point $x_{t+1}$. We can then write the inverse of the new sum as

$$\bar{C}_{t+1}^{-1} := \left(\bar{C}_t + \sum_{\ell=1}^d \left(\dot{\phi}^\ell_{x_{t+1}}\left(\dot{\phi}^\ell_{x_{t+1}}\right)^\top\right)\right)^{-1}.$$

This is the inverse of the rank-$d$ perturbed previous matrix $\bar{C}_t$. We can therefore construct this inverse using $d$ successive applications of the Sherman-Morrison-Woodbury formula for rank-one updates, each using $\mathcal{O}(m^2)$ computation. Since $\bar{C}_t$ is positive definite[6], we can represent its inverse as a numerically much more stable Cholesky factorisation $\bar{C}_t = \bar{L}_t\bar{L}_t^\top$. It is also possible to perform

[6] $C$ is the empirical covariance of the feature derivatives $\dot{\phi}^\ell_{x_i}$.

cheap rank-$d$ updates of such Cholesky factors[7]. Denote by $\bar{b}_t$ the sum of the moving average $b$. We solve (13) as

$$\hat{\theta} = C^{-1}b = \left(\frac{1}{t}\bar{C}_t\right)^{-1}\left(\frac{1}{t}\bar{b}_t\right) = \bar{C}_t^{-1}\bar{b}_t = \bar{L}_t^{-\top}\bar{L}_t^{-1}\bar{b}_t,$$

using cheap triangular back-substitution from $\bar{L}_t$, and never storing $\bar{C}_t^{-1}$ or $\bar{L}_t^{-1}$ explicitly.

Using such updates, the computational costs for updating the approximate infinite dimensional exponential family model in *every* iteration of the Markov chain are $\mathcal{O}(dm^2)$, which *constant in t*. We can therefore use *all* points in the history for constructing a proposal. See our implementation for further details.

**Algorithmic Description:**

1. Update sums

$$\bar{b}_{t+1} = \bar{b}_t - \sum_{\ell=1}^{d} \ddot{\phi}^{\ell}_{x_{t+1}} \quad \text{and} \quad \bar{C}_{t+1} = \bar{C}_t + \frac{1}{2}\sum_{\ell=1}^{d} \dot{\phi}^{\ell}_{x_{t+1}}(\dot{\phi}^{\ell}_{x_{t+1}})^{\top}.$$

2. Perform rank-$d$ update to obtain updated Cholesky factorisation $\bar{L}_{t+1}\bar{L}_{t+1}^{T} = \bar{C}_{t+1}$.
3. Update approximate infinite dimensional exponential family parameters

$$\hat{\theta} = \bar{L}_{t+1}^{-\top}\bar{L}_{t+1}^{-1}\bar{b}_{t+1}.$$

# C Ergodicity of KMC lite

**Notation** Denote by $\alpha(x_t, x^*(p'))$ the probability of accepting a $(p', x^*)$ proposal at state $x_t$. Let $a \wedge b = \min(a, b)$. Define $c(x^{(0)}) := L\epsilon^2\nabla \log \pi(x^{(0)})/2 + \epsilon^2 \sum_{i=1}^{L-1}(L-i)\nabla \log \pi(x^{(i\epsilon)})$ and $d(x^{(0)}) := \epsilon(\nabla f(x^{(0)}) + \nabla f(x^{(L\epsilon)}))/2 + \epsilon \sum_{i=1}^{L-1} \nabla f(x^{(i\epsilon)})$, where $x^{(i\epsilon)}$ is the $i$-th point of the leapfrog integration from $x = x^{(0)}$.

**Proof of Proposition 3**

*Proof.* We assumed $\pi(x)$ is log-concave in the tails, meaning $\exists x_U > 0$ s.t. for $x^* > x_t > x_U$, we have $\pi(x^*)/\pi(x_t) \leq e^{-\alpha_1(\|x^*\|_2 - \|x_t\|_2)}$ and for $x_t > x^* > x_U$, we have $\pi(x^*)/\pi(x_t) \geq e^{-\alpha_1(\|x^*\|_2 - \|x_t\|_2)}$, and a similar condition holds in the negative tail. Furthermore, we assumed fixed HMC parameters: $L$ leapfrog steps of size $\epsilon$, and wlog the identity mass matrix $I$. Following [22, 30], it is sufficient to show

$$\limsup_{\|x_t\|_2 \to \infty} \int \left[e^{s(\|x^*(p')\|_2 - \|x_t\|_2)} - 1\right] \alpha(x_t, x^*(p'))\mu(dp') < 0,$$

for some $s > 0$, where $\mu(\cdot)$ is a standard Gaussian measure. Denoting the integral $I_{-\infty}^{\infty}$, we split it into

$$I_{-\infty}^{-x_t^\delta} + I_{-x_t^\delta}^{x_t^\delta} + I_{x_t^\delta}^{\infty},$$

for some $\delta \in (0, 1)$. We show that the first and third terms decay to zero whilst the second remains strictly negative as $x_t \to \infty$ (a similar argument holds as $x_t \to -\infty$). We detail the case $\nabla f(x) \uparrow 0$ as $x \to \infty$ here, the other is analogous. Taking $I_{-x_t^\delta}^{x_t^\delta}$, we can choose an $x_t$ large enough that $x_t - C - L\epsilon x_t^\delta > x_U$, $-\gamma_1 < c(x_t - x_t^\delta) < 0$ and $-\gamma_2 < d(x_t - x_t^\delta) < 0$. So for $p' \in (0, x_t^\delta)$ we have

$$L\epsilon p' > x^* - x_t > L\epsilon p' - \gamma_1 \implies e^{-\alpha_1(-\gamma_1 + L\epsilon p')} \geq e^{-\alpha_1(x^* - x_t)} \geq \pi(x^*)/\pi(x_t),$$

where the last inequality comes from the log-concave tails assumption. For $p' \in (\gamma_2^2/2, x_t^\delta)$

$$\alpha(x_t, x^*) \leq 1 \wedge \frac{\pi(x^*)}{\pi(x_t)} \exp\left(p'\gamma_2/2 - \gamma_2^2/2\right) \leq 1 \wedge \exp\left(-\alpha_2 p' + \alpha_1\gamma_1 - \gamma_2^2/2\right),$$

[7] We use the open-source implementation provided at `https://github.com/jcrudy/choldate`

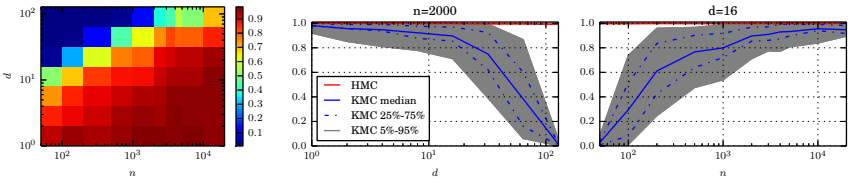


Figure 5: Acceptance probability of kernel induced Hamiltonian flow for a standard Gasussian in high dimensions for an isotropic Gaussian. **Left:** As a function of $n = m$ (x-axis) and $d$ (y-axis). **Middle:** Slices through left plot with error bars for a fixed $n = m$ and as a function in $d$ (left), and for a fixed $d$ as a function of $n = m$ (right).

where $x_t$ is large enough that $\alpha_2 = \alpha_1 L\epsilon - \gamma_2/2 > 0$. Similarly for $p' \in (\gamma_1/L\epsilon, x_t^\delta)$

$$e^{sL\epsilon p'} - 1 \geq e^{s(x^* - x_t)} - 1 \geq e^{s(L\epsilon p' - \gamma_1)} - 1 > 0.$$

Because $\gamma_1$ and $\gamma_2$ can be chosen to be arbitrarily small, then for large enough $x_t$ we will have

$$\begin{aligned} 0 < I_0^{x_t^\delta} &\leq \int_{\gamma_1/L\epsilon}^{x_t^\delta} [e^{sL\epsilon p'} - 1] \exp\left(-\alpha_2 p' + \alpha_1\gamma_1 - \gamma_2^2/2\right) \mu(dp') + I_0^{\gamma_1/L\epsilon} \\ &= e^{c_1} \int_{\gamma_1/L\epsilon}^{x_t^\delta} [e^{s_2 p'} - 1] e^{-\alpha_2 p'} \mu(dp') + I_0^{\gamma_1/L\epsilon}, \end{aligned} \tag{14}$$

where $c_1 = \alpha_1\gamma_1 - \gamma_2^2/2 > 0$ for large enough $x_t$, as $\gamma_1$ and $\gamma_2$ are of the same order. Now turning to $p' \in (-x_t^\delta, 0)$, we can use an exact rearrangement of the same argument (noting that $c_1$ can be made arbitrarily small) to get

$$I_{-x_t^\delta}^0 \leq e^{c_1} \int_{\gamma_1/L\epsilon}^{x_t^\delta} [e^{-s_2 p'} - 1] \mu(dp') < 0. \tag{15}$$

Combining (14) and (15) and rearranging as in [30, Theorem 3.2] shows that $I_{-x_t^\delta}^{x_t^\delta}$ is strictly negative in the limit if $s_2 = sL\epsilon$ is chosen small enough, as $I_0^{\gamma_2/L\epsilon}$ can also be made arbitrarily small.

For $I_{-\infty}^{-x_t^\delta}$ it suffices to note that the Gaussian tails of $\mu(\cdot)$ will dominate the exponential growth of $e^{s(\|x^*(p')\|_2 - \|x_t\|_2)}$ meaning the integral can be made arbitrarily small by choosing large enough $x_t$, and the same argument holds for $I_{x_t^\delta}^\infty$. □

# D Additional Experimental Details

This section contains additional details for the experiments in Section 6.

## D.1 Stability in High Dimensions

We reproduce the experiment in Figure 2 on an *isotropic* Gaussian in increasing dimension. As length-scales across all principal components are equal, this is a significantly less challenging target to estimate gradients for; though still useful as a benchmark representing very smooth targets. We use a standard Gaussian kernel and the same experimental protocol as for Figure 2. The estimator works slightly better than on the target considered in Figure 2, and performs well up to $d \approx 100$, see Figure 5.

## D.2 Banana target

Following [12, 10], let $X \sim \mathcal{N}(0, \Sigma)$ be a multivariate normal in $d \geq 2$ dimensions, with $\Sigma = \text{diag}(v, 1, \ldots, 1)$, which undergoes the transformation $X \to Y$, where $Y_2 = X_2 + b(X_1^2 - v)$, and $Y_i = X_i$ for $i \neq 2$. We will write $Y \sim \mathcal{B}(b, v)$. It is clear that $\mathbb{E}Y = 0$, and that

$$\mathcal{B}(y; b, v) = \mathcal{N}(y_1; 0, v)\mathcal{N}(y_2; b(y_1^2 - v), 1) \prod_{j=3}^{d} \mathcal{N}(y_j; 0, 1).$$

We choose $d = 8, V = 100$ and $b = 0.03$, which corresponds to the 'strongly twisted' 8-dimensional Banana in [12, 10]. The target is challenging due to the nonlinear dependence of the first two dimensions and the highly position dependent scaling within these dimensions.

### D.3 Pseudo-Marginal MCMC for GP Classification

**Model** Closely following [12], we consider a joint distribution of GP-latent variables $\mathbf{f}$, labels $\mathbf{y}$ (with covariate matrix $X$), and hyperparameters $\theta$, given by

$$p(\mathbf{f}, \mathbf{y}, \theta) = p(\theta)p(\mathbf{f}|\theta)p(\mathbf{y}|\mathbf{f}),$$

where $\mathbf{f}|\theta \sim \mathcal{N}(0, \mathcal{K}_\theta)$, with $\mathcal{K}_\theta$ modeling the covariance between latent variables evaluated at the input covariates: $(\mathcal{K}_\theta)_{ij} = \kappa(\mathbf{x}_i, \mathbf{x}'_j|\theta) = \exp\left(-\frac{1}{2}\sum_{d=1}^{D} \frac{(x_{i,d} - x'_{j,d})^2}{\ell_d^2}\right)$ and $\theta_d = \log \ell_d^2$. This covariance parametrisation allows to perform Automatic Relevance Determination. We here restrict our attention to the binary logistic classifier, i.e. the likelihood is given by

$$p(y_i|f_i) = \frac{1}{1 - \exp(-y_i f_i)},$$

where $y_i \in \{-1, 1\}$. Aiming for a fully Bayesian treatment, we wish to estimate the *marginal* posterior of the hyperparameters $\theta$, motivated in [4]. The marginal likelihood $p(\mathbf{y}|\theta)$ is intractable for non-Gaussian likelihoods $p(\mathbf{y}|\mathbf{f})$, but can be replaced with an unbiased estimate

$$\hat{p}(\mathbf{y}|\theta) := \frac{1}{n_{\text{imp}}} \sum_{i=1}^{n_{\text{imp}}} p(\mathbf{y}|\mathbf{f}^{(i)}) \frac{p(\mathbf{f}^{(i)}|\theta)}{q(\mathbf{f}^{(i)}|\theta)}, \tag{16}$$

where $\left\{\mathbf{f}^{(i)}\right\}_{i=1}^{n_{\text{imp}}} \sim q(\mathbf{f}|\theta)$ are $n_{imp}$ importance samples. In [4], the importance distribution $q(\mathbf{f}|\theta)$ is chosen as the Laplacian or as the Expectation Propagation (EP) approximation of $p(\mathbf{f}|\mathbf{y}, \theta) \propto p(\mathbf{y}|\mathbf{f})p(\mathbf{f}|\theta)$, leading to state-of-the-art results.

**Experimental details** We here use a Laplace approximation and $n_{\text{imp}} = 100$. We consider classification of window against non-window glass in the UCI Glass dataset, which induces a posterior that has a nonlinear shape [12, Figure 3]. Since the ground truth for the hyperparameter posterior is not available, we initially run multiple hand-tuned standard Metropolis-Hastings chains for 500,000 iterations (with a 100,000 burn-in), keep every 1000-th sample in each of the chains, and combine them. The resulting samples are used as a benchmark, to evaluate the performance all algorithms. We use the MMD between each sampler output and the benchmark sample is computed, using the polynomial kernel $(1 + \langle \theta, \theta' \rangle)^3$. This corresponds to the estimation error of all mixed moments of order up to 3.

**Cross-validation** Kernel parameters are tuned using a black box Bayesian optimisation package[8] and the median heuristic for KMC and KAMH repsectively. The Bayesian optimisation uses standard parameters and is stopped after 15 iterations, where each trial is done via a 5-fold cross-validation of the score matching objective (5). We learn parameters after MCMC 500 iterations, and then re-learn after 2000. We tried re-learning parameters after more iterations, but this did not lead to significant changes. The costs for this are neglectable in the context of PM-MCMC as estimating the marginal likelihood takes significantly more time than generating the KMC proposal.

### D.4 ABC MCMC

In this section, we give a brief background on Approximate Bayesian Computation, and how KMC can be used within the framework. We then give details of the competing approach in the final experiment in Section 6, including experimental details and an analytic counterexample.

[8] We use the open-source package `pybo`, available under `https://github.com/mwhoffman/pybo`

**Likelihood-free Models** Approximate Bayesian Computation is a method for inference in the scenario where conditional on some parameter of interest $\theta$, we can easily simulate data $x \sim f(\cdot|\theta)$, but for which the likelihood function $f$ is unavailable [6]. We however have data $y$ which assume to be from the model, and we have a prior $\pi_0(\theta)$. A simple ABC algorithm is to sample $\theta_i \sim \pi_0(\cdot)$ (or any other suitable distribution), simulate data $x_i \sim f(\cdot|\theta_i)$, and 'accept' $x_i$ as a sample from the approximate posterior $\pi_\epsilon(\theta|y)$ if $d(y, x) \leq \epsilon$. This procedure can be formalised by defining the approximate likelihood as

$$f_\epsilon(y|\theta) \propto \int g_\epsilon(y|x, \theta) f(x|\theta) \mathrm{d}x, \tag{17}$$

where $g_\epsilon(y|x, \theta)$ is an appropriate kernel that gives more importance to points for which $d(y, x)$ is smaller. In the simple case above $g_\epsilon(y|x, \theta) = \mathbf{1}_{\{d(y,x)\leq\epsilon\}}$. The ABC posterior is then found using $\pi_\epsilon(\theta|y) \propto f_\epsilon(y|\theta)\pi_0(\theta)$. Often $g_\epsilon$ is based on some low-dimensional summary statistics, which can have both advantages and disadvantages.

**Likelihood-free MCMC** There are many different way to do ABC, and clearly not all involve Markov chain Monte Carlo. If the posterior however is not similar to the prior, and if $\theta$ is more than three or four dimensional, MCMC is a sensible option. Since the likelihood (17) is intractable, typically algorithms are considered for which an approximation to either the likelihood, or the ABC posterior are used either in constructing proposals, defining Metropolis-Hastings acceptance rates, or both. We focus here on samplers which target $\pi_\epsilon(\theta|y)$ directly, c.f. [5].

**Pseudo-Marginal Metropolis-Hastings** Similar to the approach taken in Section D.3, we here accept proposals $\theta' \sim Q(\theta, \cdot)$ where $Q$ is some proposal mechanism (i.e. KMC), via replacing the likelihood with an unbiased estimate. We accept according to the ratio

$$\tilde{\alpha}(\theta, \theta') = \frac{\tilde{\pi}_\epsilon(\theta'|y)Q(\theta|\theta')}{\tilde{\pi}_\epsilon(\theta|y)Q(\theta'|\theta)}, \tag{18}$$

where $\tilde{\pi}_\epsilon(\theta|y) = \pi_0(\theta)\tilde{g}_\epsilon(y|\theta)$, and

$$\tilde{g}_\epsilon(y|\theta) = \frac{1}{n_{\text{lik}}} \sum_i g_\epsilon(y|x_i, \theta), \;\; \{x_i\}_{i=1}^{n_{\text{lik}}} \sim f(\cdot|\theta_i)$$

is a simple Monte Carlo estimator for the intractable likelihood (17). Since it is easy to simulate from $f$ then $\tilde{g}_\epsilon(y|\theta)$ is typically easy to compute. As with other general Pseudo-Marginal schemes, and as mentioned below the KMC acceptance (3), it is crucial that if $\theta'$ is accepted, the same estimate for $\tilde{\pi}(\theta'|y)$ is used on the denominator of the Hastings ratio in future iterations until the next proposal is accepted for the scheme, c.f. [3, Table 1].

We can directly adapt KMC to the ABC case via plugging in the estimated likelihood $\tilde{g}_\epsilon$ in the KMC acceptance ratio (3).

**Synthetic Likelihood Metropolis-Hastings** Following [26], one idea to approximate the intractable likelihood is to draw $n_{\text{lik}}$ samples $x_i \sim f(\cdot|\theta_i)$, and fit a Gaussian approximation to $f$, producing estimates $\hat{\mu}$ and $\hat{\Sigma}$ for the mean and covariance using $\{x_i\}_{i=1}^{n_{\text{lik}}}$. If the error functon $g_\epsilon$ is also chosen to be a Gaussian (with mean $y$ and variance $\epsilon$), then the marginal likelihood $f_\epsilon(y|\theta)$ can be approximated as

$$y|\theta \sim \mathcal{N}\left(\hat{\mu}, \hat{\Sigma} + \epsilon^2 I\right)$$

The likelihood is essentially approximated by a Gaussian $f_G$, producing a synthetic posterior $\pi_s(\cdot)$, which is then used in the accept-reject step. Clearly some approximation error is introduced by the Gaussian likelihood approximation step, but as shown in [26], it can be a reasonable choice for some models.

**Hamiltonian ABC** Introduced in [8], the synthetic likelihood formulation is used to construct a proposal, with the accept-reject step removed altogether. Hamiltonian dynamics use the gradient $\nabla \log \pi(\theta)$ to suggest candidate values for the next state of a Markov chain which are far from the current point, thus increasing the chances that the chain mixes quickly. Here the gradient of the log-likelihood is unavailable, so is approximated with that of a Gaussian (since the map $\theta \to (\mu, \Sigma)$

is not always clear this is done numerically, using a stochastic finite differences estimate of the gradient, SPAS [8, Sec. 4.3, 4.4]), giving

$$\nabla \log \pi(\theta) \approx \sum_{i=1}^{n_{\text{lik}}} \nabla \log f_G(y_i|\hat{\mu}, \hat{\Sigma}) + \nabla \log \pi_0(\theta).$$

Since there is no accept-reject step, the synthetic posterior is also the target of this scheme (although there is also further bias introduced by discretisation error), but the introduction of gradient-based dynamics is hoped to improve mixing and hence efficiency of inferences compared to random-walk type schemes.

**A Counter-example** We give a very simple toy model to highlight the bias introduced by the Hamiltonian ABC sampler. Consider posterior inference for the mean parameter in a log-Normal model. Specfically, the true model is

$$\begin{aligned} \mu &\sim \mathcal{N}(\mu_0, \tau_0), \\ y|\mu, \tau &\sim \log \mathcal{N}(\mu, \tau), \end{aligned}$$

where the precision $\tau$ and hyperparameters $\mu_0, \tau_0$ are known. The model is in fact conjugate, giving a Gaussian posterior

$$\mu|y \sim \mathcal{N}\left(\frac{\tau_0\mu_0 + \tau\sum_i \log x_i}{\tau_0 + n\tau}, \tau_0 + n\tau\right).$$

If we introduce a Gaussian approximation to the likelihood, then the mean and precision of this approximation $f_G$ are (empirical estimates for)

$$\mu_G = e^{\mu+1/2\tau}, \;\; \tau_G = 1/\text{Var}[Y_i] = \frac{e^{-2\mu-1/\tau}}{e^{1/\tau}-1},$$

which depend on the current value for $\mu$ in the chain. The resulting synthetic posterior is no longer tractable, but since it is one dimensional we can approximate it numerically. Using $\mu_0 = 0$, $\tau_0 = 1/100$, $\epsilon = 0.1$ and $\tau = 1$ then the true and approximate posteriors for 100 data points generated using the truth $\mu = 2$ are shown in Figure 6 (left). This is a proof of concept that a likelihood with a positive skew being approximated by a Gaussian introduces an upwards bias to the posterior.

**Experimental details** The simulation study in Section 6 uses a slightly more complex and multi-dimensional simulation example: a 10-dimensional multivariate skew-Normal distribution, given by

$$p(y|\theta) = 2\mathcal{N}(\theta, I)\,\Phi(\langle \alpha, y\rangle)$$

with $\theta = \alpha = \mathbf{1}\cdot 10$. In each iteration of KMC, the likelihood is estimated via simulting $n_{\text{lik}} = 10$ samples from the above likelihood. We use the mean of all samples as summary statistic, and a Gaussian similarity kernel $g_\epsilon(y|x,\theta)$ with a fixed $\epsilon = 0.55$. Both KMC and HABC use a standard Gaussian momentum, a uniformly random stepsize in $[0.01, 0.1]$ and $L = 50$ leapfrog steps. HABC is used with the suggested 'sticky random numbers' [8, Section 4.4], i.e. we use the same seed for all simulations along a single porposal trajectory. Both algorithms are run for $200 + 5000$ MCMC iterations. KMC then attempts to re-learn smoothness parameters, and stops adaptation. Burn-in samples are discarded when quantifying performance of all algorithms.

**Friction, mixing, and number of simulations** HABC is used in its 'stochastic gradient' [7] and has a 'friction' parameter that we estimate using a running average of the global covariance of all SPAS gradient evaluations, [8, Equation 21]. Note that we ran HABC with both the friction term included and removed, where we found that adding friction has severely negative impact on mixing, where not adding friction results in a wider posterior (with the same bias). Figure 4 (middle, right) show the results *without* friction, Figure 6 shows the same plots with friction. We refer to our implementation for further details.

Due to the gradient estimation in every of the $L = 50$ leap-frog steps, e*very* MCMC proposal for HABC requires $2L = 100$ simulations to be generated. In contrast, KMC only requires a single simulation, for evaluating the accept/reject probability (3). We leave studying the exact trade-offs of KMC's learning phase and its ability to mix well as compared to HABC to future work.

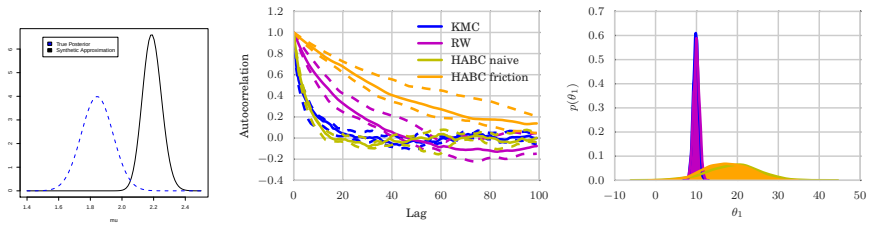


Figure 6: **Left:** Counter example showing posterior and its synthetic approximation for a simple toy model **Middle/right:** The same results as in Figure 4, but here we also show performance of HABC with added friction, which has a severely negative impact on mixing.