# Hamiltonian ABC

**Edward Meeds**
Informatics Institute
University of Amsterdam
tmeeds@gmail.com

**Robert Leenders**
Informatics Institute
University of Amsterdam
leenders.robert@gmail.com

**Max Welling** *
Informatics Institute
University of Amsterdam
welling.max@gmail.com

## Abstract

Approximate Bayesian computation (ABC) is a powerful and elegant framework for performing inference in simulation-based models. However, due to the difficulty in scaling likelihood estimates, ABC remains useful for relatively low-dimensional problems. We introduce Hamiltonian ABC (HABC), a set of likelihood-free algorithms that apply recent advances in scaling Bayesian learning using Hamiltonian Monte Carlo (HMC) and stochastic gradients. We find that a small number forward simulations can effectively approximate the ABC gradient, allowing Hamiltonian dynamics to efficiently traverse parameter spaces. We also describe a new simple yet general approach of incorporating random seeds into the state of the Markov chain, further reducing the random walk behavior of HABC. We demonstrate HABC on several typical ABC problems, and show that HABC samples comparably to regular Bayesian inference using true gradients on a high-dimensional problem from machine learning.

## 1 INTRODUCTION

In simulation-based science, models are defined by a simulator and its parameters. These are called *likelihood-free* models because, in contrast to probabilistic models, their likelihoods are either intractable to compute or must be approximated by simulations. To perform inference in likelihood-free models, a broad class of algorithms called Approximate Bayesian Computation [3, 13, 20, 12] are employed.

At the core of every ABC algorithm is simulation. To evaluate the quality of a parameter vector $\boldsymbol{\theta}$, a simulation is run using $\boldsymbol{\theta}$ as inputs and producing outputs $\mathbf{x}$. If the pseudo-data $\mathbf{x}$ is "close" to observations $\mathbf{y}$, then $\boldsymbol{\theta}$ is kept as a sample from the approximate posterior. Parameters $\boldsymbol{\theta}$ are then adjusted, depending upon the algorithm, to obtain the next sample.

In ABC, there is a fundamental trade-off between the computation required to obtain independent samples and the approximation to the true posterior. If the parameter measuring closeness is too small, then samplers "mix" poorly; on the other hand, if it is too large, then the approximation is poor. As the dimension of the parameters grows, the problem worsens, just as it does for general Bayesian inference with probabilistic models, but it is more acute for ABC due to its simulation requirement. There is therefore a deep interest in improving the efficiency of ABC samplers (in terms of computation per independent sample). In this paper we address this issue directly by using Hamiltonian dynamics to approximately sample from likelihood-free models with high-dimensional parameters.

Hamiltonian Monte Carlo (HMC) [7, 16] is perhaps the only Bayesian inference algorithm that scales to high-dimensional parameter spaces. The core computation of HMC is the gradient of the log-likelihood. Two problems arise if we consider HMC for ABC: one, how can the gradients be computed for high-dimensional likelihood-free models, and two, given a stochastic approximation to the gradient, can a valid HMC algorithm be derived?

To answer the latter, we turn to recent developments in scaling Bayesian inference using HMC and stochastic gradients [25, 5, 6]. We call these *stochastic gradient Hamiltonian dynamics* (SGHD) algorithms. SGHD algorithms are computationally efficient for two reasons. First, they avoid computing the gradient of the log-likelihood over the entire data set, instead approximating it using small batches of data, i.e. computing stochastic gradients. Second, they can maintain reasonable approximations to the Hamiltonian dynamics and therefore avoid a Metropolis-Hastings correction step involving the full data set. Different strategies are employed to do this: small step-sizes combined with Langevin dynamics [25] (stochastic gradi-

[*] Donald Bren School of Information and Computer Sciences University of California, Irvine, and Canadian Institute for Advanced Research.

ent Langevin dynamics—SGLD), using friction to prevent accumulation of errors in the Hamiltonian [5] (stochastic gradient HMC—SGHMC), and using a thermostat to control the temperature of the Hamiltonian [6] (stochastic gradient Nose-Hoover thermostats—SGNHT). Each of these strategies can be used by HABC.

In HABC, we use forward simulations to approximate the likelihood-free gradient. The key difference between SGHD methods and HABC is that the stochasticity of the gradient does not come from approximating the full data gradient with a mini-batch gradient, but by the stochasticity of the simulator. It is therefore not the expense of the simulator (though this could very well be the case for many interesting simulation-based models – see Section 7) that requires an approximation to the gradient, but the likelihood-free nature of the problem.

There are several difficulties in estimating gradients of likelihood-free models that we address with HABC. The first is due to the form of the ABC log-likelihood. As we show in Section 2, using a conditional model for $\pi(\mathbf{x}|\boldsymbol{\theta})$ provides an estimate of the ABC likelihood that is less sensitive to $\boldsymbol{\epsilon}$ and therefore is more conducive to stochastic gradient computations. The second difficulty is that for high-dimensional parameter spaces, computing the gradients naively (i.e. by finite differences (FD) [9]) can squash the gains brought by the Hamiltonian dynamics. Fortunately, we can use existing stochastic approximation algorithms [21, 22] that can be used to compute unbiased estimators of the gradient with a small number of forward simulations that is *independent* of the parameter dimension. The *stochastic perturbation stochastic approximation* (SPSA) [21] is described in Section 4

A further innovation of this paper is the use of persistent random numbers (PRNs) to improve the efficiency of the Hamiltonian dynamics. The idea behind PRNs is to use the same set of random seeds for estimating a gradient by FD or SPSA, i.e. when simulating $\pi(\mathbf{x}|\theta + d\theta)$ and $\pi(\mathbf{x}|\theta - d\theta)$ use the same random seeds. This was applied successfully to SPSA [10] (and is analogous to using the same mini-batch in stochastic gradient methods). We extend and simplify this approach by including the random seeds $\omega$ into the state of the Markov chain; by keeping the random seeds fixed for several consecutive steps, the second order gradient stochasticity is greatly reduced. We show that doing this produces a valid MCMC procedure. This approach is not exclusive to HABC; our experiments show it also helps random-walk ABC-MCMC.

We briefly review ABC in Section 2. In Section 3 we review three approaches to stochastic gradient inference using Hamiltonian dynamics: SGLD, SGHMC, and SGNHT. We then introduce Hamiltonian ABC in Section 4, where we will show how to improve the stability of the gradient estimates by using PRNs and local density estimators of the simulator. Extensions to high-dimensional parameter spaces are also discussed. In Section 5 we show how HABC behaves on a simple one-dimensional problem, then in Section 6 we compare HABC with ABC-MCMC for two problems: a low-dimensional model of chaotic population dynamics and a high-dimensional problem.

## 2 APPROXIMATE BAYESIAN COMPUTATION

Consider the Bayesian inference task of either drawing samples from or learning an approximate model of the following (usually intractable) posterior distribution:

$$\pi(\boldsymbol{\theta}|\mathbf{y}_1, \ldots, \mathbf{y}_N) \propto \pi(\boldsymbol{\theta})\pi(\mathbf{y}_1, \ldots, \mathbf{y}_N|\boldsymbol{\theta}) \qquad (1)$$

where $\pi(\boldsymbol{\theta})$ is a prior distribution over parameters $\boldsymbol{\theta} \in \mathbb{R}^D$ and $\pi(\mathbf{y}_1, \ldots, \mathbf{y}_N|\boldsymbol{\theta})$ is the likelihood of $N$ data observations, where $\mathbf{y}_i \in \mathbb{R}^J$. In ABC, the vector of $J$ observations are typically informative statistics of the raw observations. It can be shown that if the statistics used in the likelihood function are sufficient, then these algorithms sample correctly from an approximation to the true posterior [12]. The simulator is treated as a generator of random pseudo-observations, i.e. $\mathbf{x} \overset{\text{sim}}{\sim} \pi(\mathbf{x}|\boldsymbol{\theta})$ is a draw from the simulator. Discrepancies between the simulator outputs $\mathbf{x}$ and the observations $\mathbf{y}$ are scaled by a closeness parameter $\boldsymbol{\epsilon}$ and treated as likelihoods. This is the equivalent to putting an $\boldsymbol{\epsilon}$-kernel around the observations, and using a Monte Carlo estimate of the likelihood using $S$ draws of $\mathbf{x}$:

$$\pi_{\boldsymbol{\epsilon}}(\mathbf{y}|\boldsymbol{\theta}) = \int \pi_{\boldsymbol{\epsilon}}(\mathbf{y}|\mathbf{x})\pi(\mathbf{x}|\boldsymbol{\theta})d\mathbf{x} \approx \frac{1}{S}\sum_{s=1}^{S} \pi_{\boldsymbol{\epsilon}}(\mathbf{y}|\mathbf{x}^{(s)}) \qquad (2)$$

In ABC Markov chain Monte Carlo (MCMC) [13, 26] the Metropolis-Hastings (MH) proposal distribution is composed of the product of the proposal for the parameters $\boldsymbol{\theta}$ and the proposal for the simulator outputs:

$$q(\boldsymbol{\theta}', \mathbf{x}^{(1)'}, \ldots, \mathbf{x}^{(S)'}|\boldsymbol{\theta}) = q(\boldsymbol{\theta}'|\boldsymbol{\theta}) \prod_{s} \pi(\mathbf{x}^{(s)'}|\boldsymbol{\theta}') \qquad (3)$$

Using this form of the proposal distribution, and using the Monte Carlo approximation eq 2, we arrive at the following Metropolis-Hastings accept-reject probability,

$$\alpha = \min\left(1, \frac{\pi\left(\boldsymbol{\theta}'\right) \sum_{s=1}^{S} \pi_{\boldsymbol{\epsilon}}(\mathbf{y}|\mathbf{x}^{(s)'})q(\boldsymbol{\theta}|\boldsymbol{\theta}')}{\pi\left(\boldsymbol{\theta}\right) \sum_{s=1}^{S} \pi_{\boldsymbol{\epsilon}}(\mathbf{y}|\mathbf{x}^{(s)})q(\boldsymbol{\theta}'|\boldsymbol{\theta})}\right) \qquad (4)$$

If the simulations are part of the Markov chain, the algorithm corresponds to the pseudo-marginal (PM) sampler [2], otherwise it is a marginal sampler [13, 20]. For this paper we will be interested in the PM sampler because this is equivalent to having the random states that generated the simulation outputs in the state of the Markov chain, which we will use within a valid ABC sampling algorithm in Section 4.

An alternative approach to computing the ABC likelihood is to estimate the parameters of a conditional model $\pi(\mathbf{x}|\boldsymbol{\theta})$, e.g. using kernel density estimate [24] or a Gaussian model [28]. While either approach should be adequate and both have their own limits and advantages, for this paper we will use a Gaussian model. In ABC, using a conditional Gaussian model for $\pi(\mathbf{x}|\boldsymbol{\theta})$ is called a *synthetic likelihood* (SL) model [28]. For a SL log-likelihood model, we compute estimators of the first and second moments of $\pi(\mathbf{x}|\boldsymbol{\theta})$. The advantage is that for a Gaussian $\boldsymbol{\epsilon}$-kernel, we can convolve the two densities

$$\begin{aligned} \pi_{\boldsymbol{\epsilon}}(\mathbf{y}|\boldsymbol{\theta}) &= \int \mathcal{N}(\mathbf{y}|\mathbf{x}, \boldsymbol{\epsilon}^2)\mathcal{N}(\mathbf{x}|\mu_{\boldsymbol{\theta}}, \sigma^2_{\boldsymbol{\theta}})d\mathbf{x} && (5) \\ &= \mathcal{N}(\mathbf{y}|\mu_{\boldsymbol{\theta}}, \sigma^2_{\boldsymbol{\theta}} + \boldsymbol{\epsilon}^2) && (6) \end{aligned}$$

Of particular concern to this paper is the behavior of the log-likelihoods for different values of $\boldsymbol{\epsilon}$. In the $\boldsymbol{\epsilon}$-kernel case, the log-likelihood is very sensitive to small values of $\boldsymbol{\epsilon}$:

$$\begin{aligned} \log \pi_{\boldsymbol{\epsilon}}(\mathbf{y}|\boldsymbol{\theta}) &= \log \sum_s \mathcal{N}(\mathbf{y}|\mathbf{x}^{(s)}, \boldsymbol{\epsilon}^2) && (7) \\ &= \log \mathcal{N}(\mathbf{y}|\mathbf{x}^{(s)}, \boldsymbol{\epsilon}^2) + \log(1 + H) && (8) \\ &\approx -\log \boldsymbol{\epsilon} - \frac{1}{2\boldsymbol{\epsilon}^2}(\mathbf{y} - \mathbf{x}^{(m)})^2 && (9) \end{aligned}$$

where $m$ is the simulation that is closest to $\mathbf{y}$, $H$ is a sum over terms close to 0. We can see that the log-likelihood can be set arbitrarily small by decreasing $\boldsymbol{\epsilon}$. On the other hand, by using a model of the simulation at $\boldsymbol{\theta}$

$$\log \pi_{\boldsymbol{\epsilon}}(\mathbf{y}|\boldsymbol{\theta}) \approx -\frac{1}{2}\log(\sigma^2_{\boldsymbol{\theta}} + \boldsymbol{\epsilon}^2) - \frac{(\mathbf{y} - \mu_{\boldsymbol{\theta}})^2}{2(\sigma^2_{\boldsymbol{\theta}} + \boldsymbol{\epsilon}^2)} \quad (10)$$

For the SL model, $\boldsymbol{\epsilon}$ acts as a smoothing term and can be set to small values with little change to the log-likelihood, as long as the SL estimators are fit appropriately. This insensitivity to $\boldsymbol{\epsilon}$ will be used in Section 4 for estimating gradients of the ABC likelihood. Before describing HABC in full detail however, we now explain how scaling Hamiltonian dynamics in Bayesian learning can be accomplished using stochastic gradients from batched data.

# 3 SCALING BAYESIAN INFERENCE USING HAMILTONIAN DYNAMICS

Scaling Bayesian inference algorithms to massive datasets is necessary for their continuing relevance in the so-called *big data* era. We now review the role stochastic gradient methods combined with Hamiltonian dynamics have played in recent advances in scaling Bayesian inference. Most importantly, these methods have combined the ability of HMC to explore high-dimensional parameter spaces with the computational efficiency of using stochastic gradients based on small mini-batches of the full dataset. After an overview of HMC, we will briefly describe stochastic gradient Hamiltonian dynamics (SGHD), starting with using Langevin dynamics [25], then HMC with friction [5], and finally HMC with thermostats [6]. We will then make the connection between SGHD and HABC in Section 4.

## 3.1 Hamiltonian Monte Carlo

Hamiltonian dynamics are often necessary to adequately explore the target distribution of high-dimensional parameter spaces. By proposing parameters that are far from the current location and yet have high acceptance probability, Hamiltonian Monte Carlo [7, 16] can efficiently avoid random walk behavior that can render proposals in high-dimensions painfully slow to mix.

HMC simulates the trajectory of a particle along a frictionless surface, using random initial momentum $\boldsymbol{\rho}$ and position $\boldsymbol{\theta}$. The Hamiltonian function computes the energy of the system and the dynamics govern how the momentum and position change over time. The continuous Hamiltonian dynamics can be simulated by discretizing time into small steps $\eta$. If $\eta$ is small, the value of $\boldsymbol{\theta}$ at the end of a simulation can be used as proposals within the Metropolis-Hastings algorithm. Hamiltonian dynamics should propose $\boldsymbol{\theta}$ that are always accepted, but errors due to discretization may require a Metropolis-Hastings correction. It is this correction step that SGHD algorithms want to avoid as it requires computing the log-likelihood over the full data set.

More formally, the Hamiltonian $H(\boldsymbol{\theta}, \boldsymbol{\rho}) = U(\boldsymbol{\theta}) + K(\boldsymbol{\rho})$ is a function of the current potential energy $U(\boldsymbol{\theta})$ and kinetic energy $K(\boldsymbol{\rho}) = \boldsymbol{\rho}^T \boldsymbol{M}^{-1}\boldsymbol{\rho}/2$ ($\boldsymbol{M}$ is a diagonal matrix of masses which for presentation are set to 1). The potential energy is defined by the negative log joint density of the data and prior:

$$U(\boldsymbol{\theta}) = -\log \pi(\boldsymbol{\theta}) - \sum_{i=1}^{N} \log \pi(\mathbf{y}_i|\boldsymbol{\theta}) \quad (11)$$

The Hamiltonian dynamics follow

$$d\boldsymbol{\theta} = \boldsymbol{\rho} dt \qquad d\boldsymbol{\rho} = -\nabla U(\boldsymbol{\theta})dt \quad (12)$$

in simulation $dt = \eta$.

## 3.2 Stochastic Gradient Hamiltonian Dynamics

If the log-likelihood over the full data set is replaced with a mini-batch estimate, as is done for the following *stochastic gradient Hamiltonian dynamics* (SGHD) algorithms, then the error in simulating the Hamiltonian dynamics comes not only from the discretization, but from the variance of the stochastic gradient. As long as this error is controlled, either by using small steps $\eta$ (SGLD), or adding friction terms $B$ (SGHMC), or using a thermostat $\xi$ (SGNHT), the expensive MH correction step can be avoided and values of $\boldsymbol{\theta}$ from the Hamiltonian dynamics can be used as approximate samples from the posterior. SGHD algorithms belong to a larger class of *noisy Monte Carlo* methods that target intractable likelihoods; see [1] for an extensive overview of noisy Monte Carlo.

We develop SGHD from the large-scale data case, where the intractability is due to computing the full potential energy and its gradient; it is approximated using mini-batches:

$$\hat{U}(\boldsymbol{\theta}) \quad = \quad -\log \pi(\boldsymbol{\theta}) - \frac{N}{n} \sum_{i=h_1}^{h_n} \log \pi(\mathbf{y}_i|\boldsymbol{\theta}) \qquad (13)$$

$$\nabla\hat{U}(\boldsymbol{\theta}) \quad = \quad -\nabla \log \pi(\boldsymbol{\theta}) - \frac{N}{n} \sum_{i=h_1}^{h_n} \nabla \log \pi(\mathbf{y}_i|\boldsymbol{\theta}) (14)$$

where $n$ is the mini-batch size, and $h_i$ are indices chosen randomly without replacement from $[1, N]$ (i.e. it defined a random mini-batch). In likelihood-free settings, the stochasticity of the potential energy due to the mini-batches is instead caused by simulation noise; further likelihood assumptions, such as a Gaussian model, add another layer of approximation to our posterior. Below we describe three SGHD algorithms, originally developed for large-scale data applications, but for which we will apply directly to likelihood-free inference using gradient approximations in Section 4.

**Stochastic gradient Langevin dynamics** (SGLD) [25] performs one full leap-frog step of HMC. In doing so, SGLD avoids explicitly computing updates for momenta $\boldsymbol{\rho}$; the update for $\boldsymbol{\theta}$ is

$$\boldsymbol{\theta}_{t+1} \quad = \quad \boldsymbol{\theta}_t + \eta\mathcal{N}(0, \boldsymbol{I}_p) - \eta^2\nabla\hat{U}(\boldsymbol{\theta}_t)/2 \qquad (15)$$

One of the potential drawbacks of SGLD is that the momentum term is *refreshed* (implicitly) for every update of the $\boldsymbol{\theta}$, and since this means the parameter update only uses the current gradient approximation, it limits the benefits of using Hamiltonian dynamics. On the other hand, this also prevents SGLD from accumulating errors in the Hamiltonian dynamics. SGLD has been applied to another intractable likelihood model, Gibbs random fields [1], which closely resembles how SGLD is applied in this paper.

**Stochastic Gradient HMC** (SGHMC) [5] avoids $\boldsymbol{\rho}$ refreshment altogether. SGHMC makes the assumption $\nabla\hat{U}(\boldsymbol{\theta}) = \nabla U(\boldsymbol{\theta}) + \mathcal{N}(\mathbf{0}, \boldsymbol{V_\theta})$, where $\boldsymbol{V_\theta}$ is the covariance of the gradient approximation. To avoid a MH correction step at the end of a trajectory, a friction term $\boldsymbol{B}$ proportional to $\boldsymbol{V_\theta}$ is added to $\Delta\boldsymbol{\rho}$. In practice, since we can only approximate $\boldsymbol{B}$, a user defined friction term $\boldsymbol{C}$ is used. In our experiments we compute an online estimate $\hat{\boldsymbol{V}}$ and set $\boldsymbol{C} = c\boldsymbol{I}_p + \hat{\boldsymbol{V}}$.

**Stochastic Gradient thermostats** (SGNHT) [6] addresses the difficulty of estimating $\boldsymbol{B}$ by introducing a scalar variable $\xi$ who's addition to the Hamiltonian dynamics maintains the temperature of the system constant, i.e. it acts as a (Nose-Hoover) thermostat [11].

# 4 HAMILTONIAN ABC

The general approach of applying Hamiltonian dynamics to ABC requires choosing one of the SGHD algorithms and then plugging in the ABC gradient approximation $\nabla\hat{U}(\boldsymbol{\theta})$. With this in mind we leave the details of the Hamiltonian updates to previous work [25, 5, 6] and focus on the details of how stochastic gradients are computed in the likelihood-free setting. Note that in our implementation, we do not use a MH correction (except when switching seeds), though this can easily be added for any particular problem.

## 4.1 Deterministic Representations of Simulations

Implicit in each simulation run $\mathbf{x} \overset{\text{sim}}{\sim} \pi(\mathbf{x}|\boldsymbol{\theta})$ is a sequence of internally generated random numbers that are used to produce random draws from $\pi(\mathbf{x}|\boldsymbol{\theta})$. These random numbers are important to HABC because we wish to control the stochasticity of the simulator when computing its gradient. Furthermore, we will control the random numbers over multiple time steps. Instead of keeping track of random numbers, we can equivalently keep a vector of $S$ random seeds $\boldsymbol{\omega}$. This allows HABC to treat the simulation function $\pi(\mathbf{x}|\boldsymbol{\theta})$ as a blackbox, outside of which we can control the random number generator (RNG), and represent $\mathbf{x}^{(s)}$ as the output of a deterministic function; i.e. $\mathbf{x}^{(s)} = f(\boldsymbol{\theta}, \omega_s)$ instead of $\mathbf{x}^{(s)} \overset{\text{sim}}{\sim} \pi(\mathbf{x}|\boldsymbol{\theta})$. We include $\boldsymbol{\omega}$ as part of the state of our Markov chain.

## 4.2 Kernel-$\epsilon$ versus Synthetic-likelihood -based Gradients

In Section 2 we showed that the synthetic-likelihood representation of $\mathcal{L}_{\boldsymbol{\epsilon}}(\boldsymbol{\theta})$ is less sensitive to small choices of $\boldsymbol{\epsilon}$. This is particularly important to HABC as our gradient approximations are proportional to differences in $\mathcal{L}_{\boldsymbol{\epsilon}}(\boldsymbol{\theta})$; if the variance of the stochastic gradients is too high, then we must choose a very small step-size $\eta$, eliminating the usefulness of HMC for ABC. Under the deterministic representation of $\mathbf{x}^{(s)}$, we can write the log-likelihood as

$$\mathcal{L}_{\boldsymbol{\epsilon}}(\boldsymbol{\theta}) \quad \propto \quad \log \sum_s \mathcal{N}(\mathbf{y}|f(\boldsymbol{\theta}, \omega_s), \boldsymbol{\epsilon}^2) \qquad (16)$$

$$\approx \quad -\log \boldsymbol{\epsilon} - \frac{1}{2\boldsymbol{\epsilon}^2}(\mathbf{y} - f(\boldsymbol{\theta}, \omega_m))^2 \qquad (17)$$

In the second line we have assumed $\boldsymbol{\epsilon}$ is very small and $m$ is the index of the random seed producing the closest simulation to $\mathbf{y}$. For a finite difference approximation,

**Algorithm 1** $\nabla U$ SPSA-ABC

```
inputs: $\boldsymbol{\theta}, d_{\boldsymbol{\theta}}, f, \boldsymbol{\omega}, \mathcal{L}_{\boldsymbol{\epsilon}}, \pi, R$
$\hat{\boldsymbol{g}} \leftarrow \mathbf{0}$
for $r = 1 : R$ do
    $\boldsymbol{\Delta} \sim 2 \cdot$ Bernouilli $(1/2, D)$ - 1
    for $s = 1 : S$ do
        $\mathbf{x}_{+}^{(s)} \leftarrow f\left(\boldsymbol{\theta} + d_{\boldsymbol{\theta}}\boldsymbol{\Delta}, \omega_s\right)$
        $\mathbf{x}_{-}^{(s)} \leftarrow f\left(\boldsymbol{\theta} - d_{\boldsymbol{\theta}}\boldsymbol{\Delta}, \omega_s\right)$
    end for
    $\hat{\boldsymbol{g}} \leftarrow \hat{\boldsymbol{g}} + \left(\mathcal{L}_{\boldsymbol{\epsilon}}(\{\mathbf{x}_{+}^{(s)}\}) - \mathcal{L}_{\boldsymbol{\epsilon}}(\{\mathbf{x}_{-}^{(s)}\})\right) \cdot \boldsymbol{\Delta}^{-1}$
end for
$\hat{\boldsymbol{g}} \leftarrow \hat{\boldsymbol{g}}/(2d_{\boldsymbol{\theta}}R) + \nabla \log \pi(\boldsymbol{\theta})$
return $-\hat{\boldsymbol{g}}$
```

$\partial\mathcal{L}_{\boldsymbol{\epsilon}}(\boldsymbol{\theta})/\partial\boldsymbol{\theta}$ is

$$\frac{1}{4d_{\boldsymbol{\theta}}\boldsymbol{\epsilon}^2}\left((\mathbf{y} - f(\boldsymbol{\theta} - d_{\boldsymbol{\theta}}, \omega_m^{-}))^2 - (\mathbf{y} - f(\boldsymbol{\theta} + d_{\boldsymbol{\theta}}, \omega_m^{+}))^2\right) \tag{18}$$

On the other hand, the synthetic-likelihood is stable; using a deterministic representation, we have

$$\mu_{\boldsymbol{\theta}} = \frac{1}{S}\sum_s f(\boldsymbol{\theta}, \omega_s) \qquad \sigma_{\boldsymbol{\theta}}^{s} = \frac{1}{S-1}\sum_s (\mu_{\boldsymbol{\theta}} - f(\boldsymbol{\theta}, \omega_s))^2 \tag{19}$$

the gradients (for a 1-dim problem) use $\boldsymbol{\epsilon}$ as a smoothness prior in $\partial\mathcal{L}_{\boldsymbol{\epsilon}}(\boldsymbol{\theta})/\partial\boldsymbol{\theta}$:

$$-\frac{1}{2}\log\left(\frac{\sigma_{\boldsymbol{\theta}+}^2 + \boldsymbol{\epsilon}^2}{\sigma_{\theta-}^2 + \boldsymbol{\epsilon}^2}\right) - \frac{(\mathbf{y} - \mu_{\boldsymbol{\theta}+})^2}{2(\sigma_{\boldsymbol{\theta}+}^2 + \boldsymbol{\epsilon}^2)} + \frac{(\mathbf{y} - \mu_{\boldsymbol{\theta}-})^2}{2(\sigma_{\boldsymbol{\theta}-}^2 + \boldsymbol{\epsilon}^2)} \tag{20}$$

In Figure 2, as part of our demonstration of HABC, we compare the gradient approximations around the true $\boldsymbol{\theta}_{\text{MAP}}$ using SL versus kernel-$\boldsymbol{\epsilon}$ for a simple problem. Although there is a small bias using SL due to its Gaussian assumption, it has much smaller variance, convergence to this (biased) posterior should be stable. Further, [19] showed that convergence for SGHD type algorithms depends on the tails of the log-posterior, which suggests that despite its bias, the non-heavy tails of the Gaussian may allow SL to produce a more efficient Markov chain.

### 4.3 From Finite Differences to Simultaneous Perturbations

If the dimension of $\boldsymbol{\theta}$ is small, then *finite difference stochastic approximation* (FDSA) [9] can be applied to $\nabla U(\boldsymbol{\theta})$ (conditioned on random seeds $\boldsymbol{\omega}$). The number of simulations required for FDSA is $2SD$, which may be acceptable for some small ABC problems. Our main goal is to scale ABC to high-dimensions and for that we need an alternative stochastic approximation to $\nabla U(\boldsymbol{\theta})$.

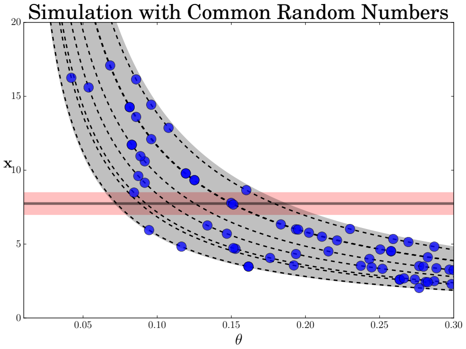

Figure 1: A view of a simulator using persistent random numbers; in other contexts, these are called common random numbers [10]. The horizontal line represents $\mathbf{y}$ and red shading $\pm 2\epsilon$. The shaded curved region represents $2\sigma$ of $\pi(\mathbf{x}|\boldsymbol{\theta})$. The dashed lines are $f(\boldsymbol{\theta}, \omega_s)$ for several values of $\omega$. The blue circles are potential random samples from $\pi(\mathbf{x}|\boldsymbol{\theta})$. For a fixed value $\omega_s$, the simulator produces deterministic outputs that change smoothly, even though the simulator itself is quite noisy.

In the gradient-free setting, Spall [21, 22] provides a stochastic approximate to the true gradient using only 2 forward simulations for any dimension $D$ (though the approximation can be improved by averaging $R$ estimates). Spall's *simultaneous perturbation stochastic approximation* (SPSA) algorithm works as follows. Let $L$ be the gradient-free function we wish to optimize. Each approximation randomly generates a *perturbation mask* (our name) $\boldsymbol{\Delta}$ of dimension $D$ where entry $\boldsymbol{\Delta}_d \sim 2$Bernouilli$(1/2)-1$ (i.e. all entries randomly set to $\pm1$). Then $L$ is evaluated at $\boldsymbol{\theta} + d_{\boldsymbol{\theta}}\boldsymbol{\Delta}$ and $\boldsymbol{\theta} - d_{\boldsymbol{\theta}}\boldsymbol{\Delta}$, giving the gradient approximation $\hat{\boldsymbol{g}}(\boldsymbol{\theta}) \approx \partial L(\boldsymbol{\theta})/\partial\boldsymbol{\theta}$:

$$\hat{\boldsymbol{g}}(\boldsymbol{\theta}) = \frac{L\left(\boldsymbol{\theta} + d_{\boldsymbol{\theta}}\boldsymbol{\Delta}\right) - L\left(\boldsymbol{\theta} - d_{\boldsymbol{\theta}}\boldsymbol{\Delta}\right)}{2d_{\boldsymbol{\theta}}}\begin{bmatrix} 1/\Delta_1 \\ 1/\Delta_2 \\ \vdots \\ 1/\Delta_D \end{bmatrix} \tag{21}$$

If we let $\hat{\boldsymbol{g}}_r(\boldsymbol{\theta})$ be the estimate using perturbation mask $\boldsymbol{\Delta}_r$, the estimate $\hat{\boldsymbol{g}}(\boldsymbol{\theta})$ can be improved by averaging $\hat{\boldsymbol{g}}(\boldsymbol{\theta}) = 1/R\sum_r \hat{\boldsymbol{g}}_r(\boldsymbol{\theta})$. Algorithm 1 shows SPSA to estimate $\nabla U(\boldsymbol{\theta})$. The number of simulations required for SPSA is $2SR$, where $R \geq 1$.

Variations of SPSA include *one-sided* SPSA [22] (we use what Spall calls 2SPSA) and an algorithm for estimating the Hessian based on the same principle as SPSA [23]. The one-sided version is attractive computationally, but for HABC, the updates for $\boldsymbol{\theta}$ require simulating two-sides anyway (once at $\boldsymbol{\theta}$, after a step is taken, and once for the one-sided gradient). SPSA has also been used within a procedure for maximum-likelihood estimation for hidden Markov models using ABC [8].

### 4.4 Persistent Random Numbers

The usefulness of applying *persistent* random numbers (PRNs) in SPSA has been previously demonstrated [10]. In that work, the same random numbers are used to simulate both sides of the optimization function within the SPSA gradient. This makes sense intuitively, as we would generally assume that the expected simulation function varies smoothly in $d\boldsymbol{\theta}$; by using PRNs, this smoothness is easily exploited (see Figure 1). If we were to apply SPSA to Bayesian learning, then using PRNs in the gradient step would be analogous to using the same mini-batch for both sides of the computation. In the case where the number of random numbers is unknown or is itself random, we can simply consider seeds of the random number generator instead of vectors of random numbers.

In addition to using PRNs in simulations for each gradient computation, we have found that using PRNs helps HABC explore the parameter landscape more easily for some algorithms and problems. Intuitively, for a gradient-based sampling algorithm, it means a particle can slide along a smooth Hamiltonian landscape because the additive noise is suppressed. This is very similar to using dependent random streams to drive MCMC [15, 17], the main difference we believe is that we are using the Hamiltonian dynamics to drive proposals for $\boldsymbol{\theta}$ and using persistent seeds $\boldsymbol{\omega}$ to suppress simulation noise. The full benefits of suppressing the noise may be limited, however. Recent work has shown that scaling HMC for large data applications may be fundamentally limited [4]: noise from mini-batches causes biases in trajectories, which require either increasing mini-batch sizes (in our case, running more simulations) or decreasing the step size.

Using random seeds (versus, say, a set of random numbers) allows us to treat the simulator as a black-box, setting the random seed of its RNG without knowing the internal mechanisms it uses to generate random numbers. In light of our arguments above, we propose including persistent random seeds $\boldsymbol{\omega}$ in the state of our Markov chain. We will now describe a simple Metropolis-Hastings transition operator that randomly proposes *flipping* each seed $\omega_s$ at time $t$ with some probability $\gamma$.

This Metropolis-Hastings transition conditions of the current parameter location $\boldsymbol{\theta}$ and proposes changing a single random seed $\omega$ (it easily generalizes to $S$ seeds). The procedure is as follows: 1) propose a new seed $\omega^{'} \sim q(\omega^{'}|\omega) = \pi(\omega)$ (independent of the current seed and from its uniform prior); 2) simulate *deterministically* $\mathbf{x}^{'} = f(\boldsymbol{\theta}, \omega^{'})$; 3) compute the acceptance ratio (which reduces to the ratio of $\pi(\mathbf{y}|\mathbf{x}^{'})/\pi(\mathbf{y}|\mathbf{x})$). It is straightforward to show that this leaves the target distribution invariant. The probability of the proposal is $q(x^{'}, \omega^{'}|\boldsymbol{\theta}, \omega) = \pi(\omega^{'})\delta(\mathbf{x}^{'} - f(\boldsymbol{\theta}, \omega^{'}))$, where $\delta(a)$ is a delta function at $a = 0$. Because the $q$ has this form, the acceptance ratio simplifies:

$$\frac{\pi_{\boldsymbol{\epsilon}}(\mathbf{y}|\mathbf{x}^{'})\pi(\omega^{'})\pi(\mathbf{x}^{'}|\boldsymbol{\theta}, \omega_{'})\pi(\omega)\delta(\mathbf{x} - f(\boldsymbol{\theta}, \omega))}{\pi_{\boldsymbol{\epsilon}}(\mathbf{y}|\mathbf{x})\pi(\omega)\pi(\mathbf{x}|\boldsymbol{\theta}, \omega)\pi(\omega^{'})\delta(\mathbf{x}^{'} - f(\boldsymbol{\theta}, \omega^{'}))} = \frac{\pi_{\boldsymbol{\epsilon}}(\mathbf{y}|\mathbf{x}^{'})}{\pi_{\boldsymbol{\epsilon}}(\mathbf{y}|\mathbf{x})} \tag{22}$$

In pseudo-marginal ABC-MCMC one could propose $q(\mathbf{x}^{'(s)}|\boldsymbol{\theta})$ (fixing $\boldsymbol{\theta}$) and still sample correctly from the distribution of simulations with high likelihood at $\boldsymbol{\theta}$. What we propose is slightly different. By instead keeping the random seeds fixed, we can sample $\boldsymbol{\theta}$ using HABC and use $\boldsymbol{\omega}$ as PRNs within the gradient computation step and suppress gradient noise over time. In this way, random seeds carry over the same additive noise from one step to the next.

## 5 Demonstration

We use a simple $D = 1$ problem to demonstrate HABC. Let $y = \frac{1}{N}\sum_i e_i$, where $e_i \sim \text{Exp}(1/\boldsymbol{\theta}^{\star})$; $\boldsymbol{\theta}^{\star} = 0.15$, $N = 20$, and $y = 7.74$ in our concrete example. Assuming $\pi(\theta) = \text{Gamma}(\alpha, \beta)$, the true posterior is a gamma distribution with shape $\alpha + N$ and rate $\beta + Ny$. Our simulator therefore generates the average of $N$ exponential random variates with rate $\lambda = 1/\theta$. Data $x \overset{\text{sim}}{\sim} \pi(x|\theta)$ are shown in Figure 1. We have explicitly shown the smoothness of the simulator by generating data along trajectories of fixed seeds $\omega_s$; i.e. for several $\omega_s$ we vary $\theta$ (dashed lines are function $f(\theta, \omega_s)$) and randomly reveal simulation data (blue circles). The horizontal line with shading indicates $y \pm 2\epsilon$, where $\epsilon = 0.37$ is used throughout the demonstration.

### 5.1 Bias and Variance of $\nabla\hat{U}(\boldsymbol{\theta})$

To test our assumption that the synthetic-likelihood model is better suited for HABC, we ran FDSA at the true $\theta_{\text{MAP}}$. Using $S = 5$ and $S = 50$ and fixing $\epsilon = 0.37$, we gather $10K$ gradients samples using kernel-$\boldsymbol{\epsilon}$ and SL likelihoods. These gradient estimate densities are shown in Figure 2. An unbiased estimate of the gradient should be centered at 0. There are two important results. First, the SL estimates have a small bias, even at $S = 50$. This is because it is estimating the true Gamma distribution of $\pi(\mathbf{x}|\boldsymbol{\theta})$ with a Gaussian. We can analytically estimate this bias as $S \to \infty$; for this example it is $-7.8$ which is what SL estimates are centered around ($-9.3$ for $S = 5$ and $7.3$ for $S = 50$). The kernel-$\boldsymbol{\epsilon}$ likelihood, on the other hand, exhibits low bias at $S = 50$. However, the second important result is the variances. SL variances decrease quickly with $S$: $\sigma^2 = 43^2 \to 4.9^2$, whereas kernel-$\boldsymbol{\epsilon}$ starts very high and remains high: $\sigma^2 = 147^2 \to 19^2$. It is for this reason that we have chosen to use SL likelihoods for our gradient estimates, despite their small bias. As mentioned in Section 4.2 it is possible that other likelihood models, such as a kernel density estimate, might provide low bias and low variance gradient estimates. We leave this for future work.

### 5.2 Posterior Inference using HABC

We ran chains of length $50K$ for SL-MCMC, SGLD, SGHMC, and SGNHT versions of HABC using SL gra-

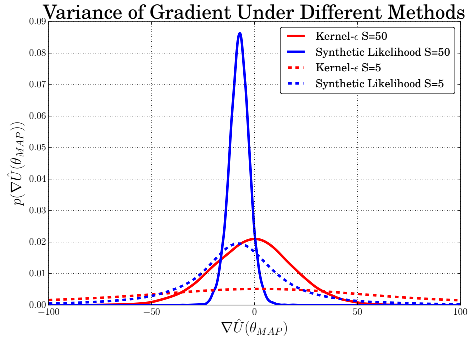

Figure 2: Variance of gradient estimation using kernel-$\epsilon$ and SL for different values of $S \in \{5, 50\}$ and fixed $\epsilon = 0.37$ (the same used in the other results). When $S = 5$, the empirical estimates of $\nabla\hat{U}(\boldsymbol{\theta}_{\text{MAP}})$ are $-12 \pm 147$ (kernel-$\epsilon$) and $-9.3 \pm 43$ (SL). When $S = 50$ they are $-0.80 \pm 19$ (kernel-$\epsilon$) and $-7.3 \pm 4.9$ (SL). Note the large discrepancy in variance. Note the limit of $S \to \infty$, $\nabla\hat{U}(\boldsymbol{\theta}_{\text{MAP}}) = -7.8$. The bias if SL gradients is due to its Gaussian approximation (smoothed by $\epsilon$) of $\pi(\mathbf{x}|\boldsymbol{\theta})$, which is a heavy-tailed Gamma distribution (the sum of $N$ exponentials).

dient estimates ($S = 5$). SL-MCMC refers to pseudo-marginal ABC-MCMC. We note that SGHMC gave results nearly identical to SGNHT, so are not shown due to space limitations. In one set of experiments, the same random seeds were used for gradient computations but did not persist over time steps; these experiments are called *non-persistent*. In another set of runs, we resampled $\omega_s$ at each time step with probability $\gamma = 0.1$; these experiments are *persistent*. In Figure 3 we show the posterior distributions for these experiments; in Table 1 we report the *total variational distance* between the true posterior and the ABC posteriors using the first $10K$ samples and after $50K$ samples (averaged over 5 chains). Of note is the poor approximation of SG-Thermostats when the seeds are not persistent. By adding persistent seeds, SGNHT gives similar posteriors to the other methods.

In Figure 4 we show the trace plots of the last 1000 samples from a single chain for each algorithm. In the left column, traces for non-persistent random seeds are shown, and on the right, traces for persistent seeds. We can observe that persistent random seeds further reduces the random walk behavior of all three methods. We also observe small improvements in total variational distance for SL-MCMC and SGLD, while SGNHT improves significantly. We find this a compelling mystery. Is it because of the interaction between hyperparameters and stochastic gradients, or is this an artifact of this simple model?

## 6 Experiments

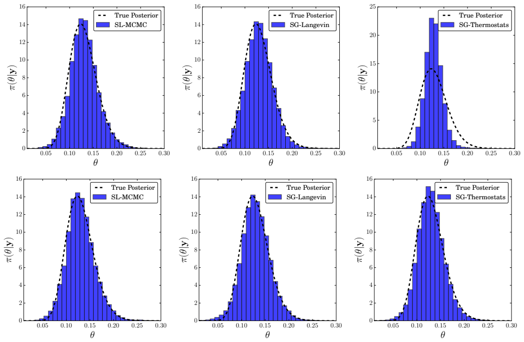

Figure 3: Posterior distributions for the demonstration problem; columns left to right: SL-MCMC, SGLD (SG-Langevin), SGNHT (SG-Thermostats). **Top row:** No persistent seeds. **Bottom row:** Persistent seeds with $\gamma = 0.1$. Histograms of the posterior estimates are overlaid with the true posterior (dashed line). All algorithms (except for SGNHT for non-persistent $\boldsymbol{\omega}$) give roughly the same posterior estimate. By adding persistent $\boldsymbol{\omega}$ SGNHT achieved similar posteriors to the other algorithms.

Table 1: Average total variational distance (tvd) for the demonstration problem. *Non-persistent* used no persistent random seeds, whereas *Persistent* randomly proposes a new $\omega_s$ with $\gamma = 0.1$. Each algorithms' parameters were optimized for minimal tvd after $10K$ samples. The results for SGHMC (not shown) and SGNHT are nearly identical.

| | Non-persistent | | Persistent | |
|---|---|---|---|---|
| Algo | $10K$ | $50K$ | $10K$ | $50K$ |
| SL-MCMC | 0.047 | 0.045 | 0.045 | 0.045 |
| SGLD | 0.049 | 0.048 | 0.048 | 0.043 |
| SGNHT | 0.232 | 0.239 | 0.055 | 0.051 |

We present experimental results comparing HABC with standard ABC-MCMC for two challenging simulators. The first is the *blowfly* model which uses stochastic differential equations to model possibly chaotic population dynamics [28]. Although it is a low-dimensional problem, the noise and chaotic behavior of the model make it challenging for gradient-based sampling. Our second experiment applies HABC to a Bayesian logistic regression model. Although we only use 2 classes (0's versus 1's), the dimensionality is very high ($D = 1568$). We show that HABC can work well despite using SPSA gradients.

### 6.1 Blowfly

For these experiments, a simulator of adult sheep blowfly populations [28] is used with statistics set to those from [14]. The observational vector $\mathbf{y}$ is a time-series of a fly population counted daily. The population dynamics are modeled using a stochastic differential equation[1]

$$N_{t+1} = PN_{t-\tau}\exp(-N_{t-\tau}/N_0)e_t + N_t\exp(-\delta\epsilon_t)$$

where $e_t \sim \mathcal{G}(1/\sigma_p^2, 1/\sigma_p^2)$ and $\epsilon_t \sim \mathcal{G}(1/\sigma_d^2, 1/\sigma_d^2)$ are sources of noise, and $\tau$ is an integer. In total, there are $D =$

[1] Equation 1 in Section 1.2.3 of the supplementary information in [28].

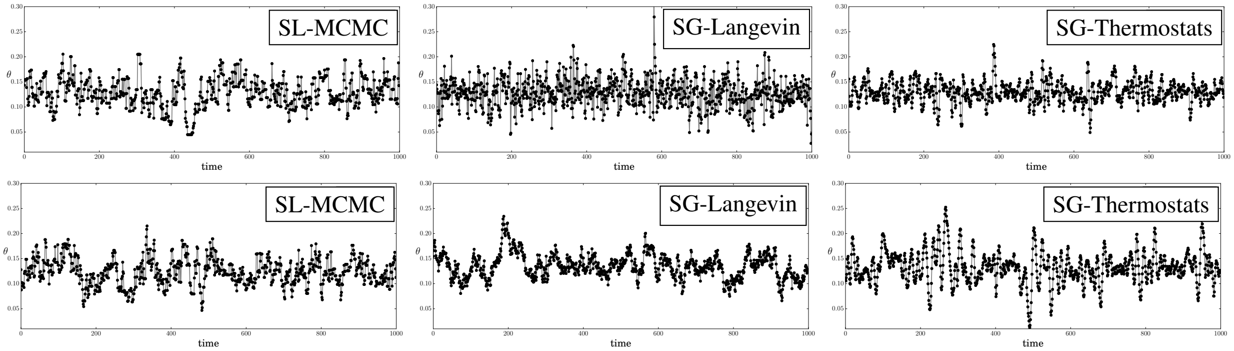

Figure 4: Trajectories of the last 1000 $\boldsymbol{\theta}$ samples for the demonstration problem. **Top row:** Non-persistent random seeds. **Bottom row:** Persistent random seeds with $\gamma = 0.1$. Each algorithm's parameters were optimized to minimize the total variational distance. With persistent seeds, each algorithm's random walk behavior is suppressed. Without persistent seeds, the optimal step-size $\eta$ for SGNHT is small, resulting in an under-dispersed estimate of the posterior; when the seeds are persistent, the gradients are more consistent, and the optimal step-size is larger and therefore there is larger injected noise. The resulting posteriors are shown in Figure 3.

6 parameters $\theta = \{\log P, \log \delta, \log N_0, \log \sigma_d, \log \sigma_p, \tau\}$. As [14] we place broad log-normal priors over $\theta_{1\ldots5}$ and a Poisson prior over $\tau$. This is considered a challenging problem because slight changes to some parameter settings can produce degenerate $\mathbf{x}$, while others settings can be very noisy due to the chaotic nature of the equations. The statistics from [14] are used ($J = 10$): the log average of 4 quantiles of $N/1000$, the average of 4 quantiles of the first-order differences in $N/1000$, and the number of maximal population peaks under two different thresholds.

We compare difference HABC algorithms with ABC-MCMC for the blowfly population problem. We use $\boldsymbol{\epsilon} = \{1/2, 1/2, 1/2, 1/2, 1/4, 1/4, 1/4, 1/4, 3/4, 3/4\}$ (slightly different $\boldsymbol{\epsilon}$ from [14]) and $S = 10$ for all experiments (this means that there are $S$ random seeds). We use SPSA with $R = 2$ using SL log-likelihoods for all HABC gradient estimates. Without persistent seeds, the number of simulations per time-step is $2SR$ (about double marginal ABC-MCMC) and with it is $2SR + 2S\gamma$.

Figure 5 show the posterior distributions for three parameters for SL-MCMC, SGLD, and SGNHT using non-persistent seeds (persistent seeds, not shown, produced very similar posteriors). In the second row we show the trajectories of two parameters, clearly showing the suppressed random walk behavior of SGLD and SG-Thermostats relative to ABC-MCMC. In Figure 6 the scatter plots of trajectories are shown for two parameters. Though not shown due to space limitations, we have found that persistent seeds can improve convergence of the posterior predictive distribution. Further experiments with persistent seeds needs to be carried out to understand the extent to which the help and how to determine when they are necessary, if at all.

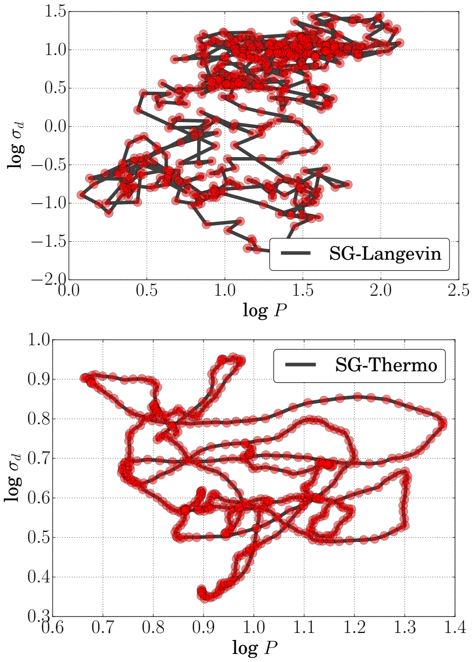

Figure 6: Blowfly trajectories of two parameters over the last 1000 time-steps. **Top:** SGLD and **Bottom**: SGNHT (SG-Thermostats). Relative to SL-MCMC (not shown), the Hamiltonian dynamics clearly show persistent $\boldsymbol{\theta}$ trajectories.

### 6.2 Bayesian Logistic Regression

We perform Bayesian inference on a logistic regression model using the digits 0 and 1 from MNIST. Although not technically an ABC problem because we use the actual likelihoods, it still represents a high-dimensional problem ($D = 1568$) and is therefore useful to evaluate the potential of SPSA-like gradients in actual HABC problems. We first ran stochastic gradient descent to determine $\boldsymbol{\theta}_{\text{MAP}}$ using the true gradient. We then ran SGLD and SGNHT starting $\boldsymbol{\theta}_{\text{MAP}}$ to discover how well the algorithms explore the posterior. We examine how SGLD and SGNHT trajectories are affected by using SPSA instead of the *true* gradients. We use $n = 100$ sized mini-batches and $R = 10$ perturbations for SPSA. Figure 7 shows samples randomly projected onto 2 dimensions (1000 evenly sub-sampled from

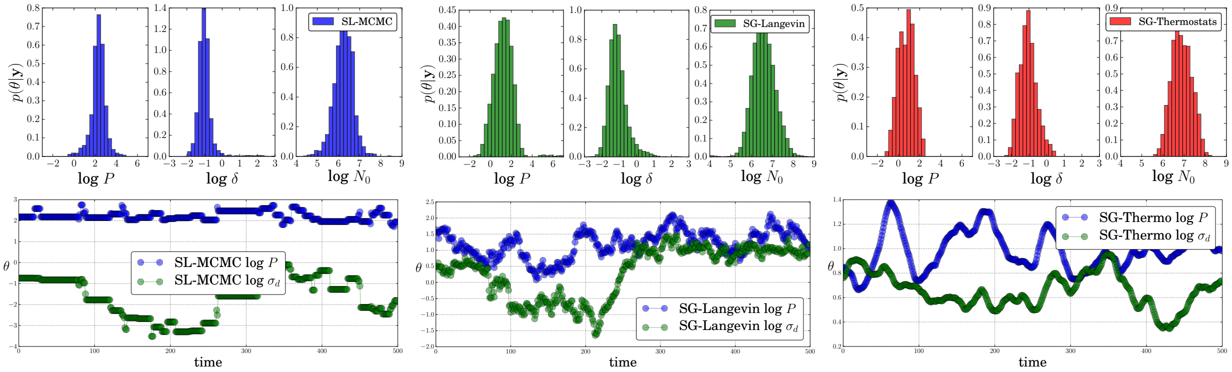

Figure 5: Blowfly posterior distributions (non-persistent seeds). **Top row**: Posteriors for three parameters for SL-MCMC (left set of three), SGLD (SG-Langevin) (middle), and SGNHT (SG-Thermostats) (right). **Bottom row:** Last trajectories of the last 1000 samples for two parameters for the same algorithms.

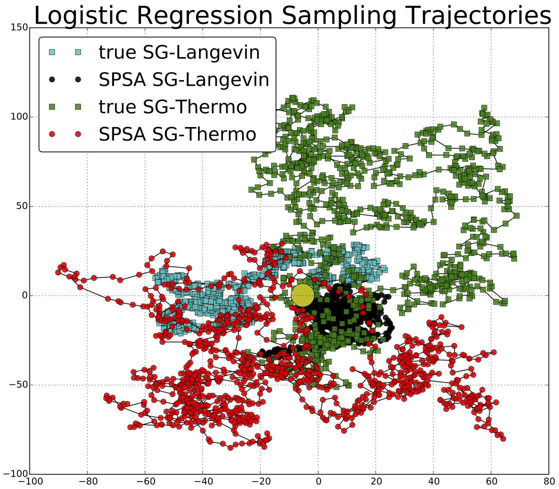

Figure 7: Bayesian logistic regression sampling trajectories randomly projected. The yellow circle is the projected MAP of $\boldsymbol{\theta}$.

$10K$). We can clearly see that the trajectories using SPSA exhibit very similar behavior to Bayesian learning with the true gradients. This is very positive result that indicates HABC can successfully exploit the noisy and less informative gradients of SPSA.

## 7 DISCUSSION AND CONCLUSION

Hamiltonian ABC proposes a new set of algorithms for Bayesian inference of likelihood-free models. HABC builds upon the connections between Hamiltonian Monte Carlo with stochastic gradients and well-established gradient approximations based on a minimal number of forward simulations, even in high-dimensions. We have performed some preliminary experiments showing the feasibility of running HABC on both small and large problems, and we hope that the door has been opened for exploration of larger simulation-based models using HABC.

Another innovation we introduce is the use of persistent random seeds to suppress the simulator noise and therefore smooth the simulation landscape over a local region of parameter space. For some algorithms run on certain models, improved performance has been observed. This is most likely to be the case for simulators with large additive noise and algorithms that benefit from long Hamiltonian trajectories (i.e. SGHMC and SGNHT). We feel that new classes of ABC algorithms could develop from using persistent random seeds, not just gradient-based samplers but traditional ABC-MCMC.

There are several unresolved and open questions regarding the application of stochastic gradients to ABC. The first issue is the importance of the bias-variance relationship for different ABC likelihood models. We found that using gradients based on the synthetic-likelihood greatly reduced their variance, but introduced a small bias, because of its Gaussian assumption. The second issue is setting algorithm parameters, in particular the step-sizes $\eta$, the injected noise $C$ (for SGHMC/SGNHT), and the number of SPSA repetitions $R$. All of these parameters are highly interactive. Can we use statistical tests during the MCMC run to determine $R$? Should $\eta$ and $C$ be set differently in the ABC setting? One final issue is monitoring or determining whether the correct amount of noise is being injected to ensure proper sampling. In SGLD [25], for example, we can always turn down $\eta$ so that the injected noise term dominates, but when our goal is efficient exploration of the posterior, this is not a very satisfying solution.

Expensive simulators are an important class of models that we do not address in this work. However, previous work in Bayesian inference has shown the usefulness of HMC-based proposals based on Gaussian process of log-likelihood surfaces [18]. We could similarly use HABC with ABC surrogate models [14, 27] to minimize simulation calls, yet still benefit from Hamiltonian dynamics.

### Acknowledgements

We thank the anonymous reviewers for the many useful comments that improved this manuscript. MW acknowledges support from Facebook, Google, and Yahoo.

# References

[1] Alquier, Pierre, Friel, Nial, Everitt, Richard, and Boland, Aidan. Noisy Monte Carlo: Convergence of Markov chains with approximate transition kernels. *Statistics and Computing*, pp. 1–19, 2014.

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

[3] Beaumont, Mark A, Zhang, Wenyang, and Balding, David J. Approximate Bayesian computation in population genetics. *Genetics*, 162(4):2025–2035, 2002.

[4] Betancourt, MJ. The Fundamental Incompatibility of Hamiltonian Monte Carlo and Data Subsampling. *Journal of Machine Learning Research*, 37, 2015.

[5] Chen, Tianqi, Fox, Emily B, and Guestrin, Carlos. Stochastic gradient Hamiltonian Monte Carlo. 2014.

[6] Ding, Nan, Fang, Youhan, Babbush, Ryan, Chen, Changyou, Skeel, Robert D, and Neven, Hartmut. Bayesian sampling using stochastic gradient thermostats. In *Advances in Neural Information Processing Systems*, pp. 3203–3211, 2014.

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

[8] Ehrlich, Elena, Jasra, Ajay, and Kantas, Nikolas. Gradient Free Parameter Estimation for Hidden Markov Models with Intractable Likelihoods. *Methodology and Computing in Applied Probability*, pp. 1–35, 2013.

[9] Kiefer, Jack, Wolfowitz, Jacob, et al. Stochastic estimation of the maximum of a regression function. *The Annals of Mathematical Statistics*, 23(3):462–466, 1952.

[10] Kleinman, Nathan L, Spall, James C, and Naiman, Daniel Q. Simulation-based optimization with stochastic approximation using common random numbers. *Management Science*, 45(11):1570–1578, 1999.

[11] Leimkuhler, Benedict and Reich, Sebastian. A metropolis adjusted Nosé-Hoover thermostat. *ESAIM: Mathematical Modelling and Numerical Analysis*, 43(04):743–755, 2009.

[12] Marin, J.-M., Pudlo, P., Robert, C.P., and Ryder, R.J. Approximate bayesian computational methods. *Statistics and Computing*, 22:1167–1180, 2012.

[13] Marjoram, Paul, Molitor, John, Plagnol, Vincent, and Tavaré, Simon. Markov chain Monte Carlo without likelihoods. *Proceedings of the National Academy of Sciences*, 100(26):15324–15328, 2003.

[14] Meeds, Edward and Welling, Max. GPS-ABC: Gaussian process surrogate approximate bayesian computation. *Uncertainty in AI*, 2014.

[15] Murray, Iain and Elliott, Lloyd T. Driving Markov chain Monte Carlo with a dependent random stream. *arXiv:1204.3187*, 2012.

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

[17] Neal, Radford M. How to View an MCMC Simulation as a Permutation, with Applications to Parallel Simulation and Improved Importance Sampling. *Technical Report No. 1201, Dept. of Statistics, University of Toronto*, 2012.

[18] Rasmussen, C.E. Gaussian processes to speed up hybrid monte carlo for expensive bayesian integrals. *Bayesian Statistics*, 7:651–659, 2003.

[19] Roberts, Gareth O and Tweedie, Richard L. Exponential convergence of Langevin distributions and their discrete approximations. *Bernoulli*, pp. 341–363, 1996.

[20] Sisson, Scott A and Fan, Yanan. Likelihood-free markov chain monte carlo. *Arxiv preprint arXiv:1001.2058*, 2010.

[21] Spall, James C. Multivariate stochastic approximation using a simultaneous perturbation gradient approximation. *Automatic Control, IEEE Transactions on*, 37(3):332–341, 1992.

[22] Spall, James C. Adaptive stochastic approximation by the simultaneous perturbation method. *Automatic Control, IEEE Transactions on*, 45(10):1839–1853, 2000.

[23] Spall, James C. Monte Carlo computation of the Fisher information matrix in nonstandard settings. *Journal of Computational and Graphical Statistics*, 14(4), 2005.

[24] Turner, Brandon M. and Sederberg, Per B. A generalized, likelihood-free method for posterior estimation. *Psychonomic Bulletin & Review*, 21(2):227–250, 2014.

[25] Welling, Max and Teh, Yee W. Bayesian learning via stochastic gradient Langevin dynamics. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pp. 681–688, 2011.

[26] Wilkinson, R. Approximate Bayesian computation (ABC) gives exact results under the assumption of model error. *Statistical Applications in Genetics and Molecular Biology*, 12(2):129–142, 2013.

[27] Wilkinson, R. Accelerating abc methods using gaussian processes. *AISTATS*, 2014.

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