Hamiltonian ABC

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 performs comparably to regular Bayesian inference 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 [2, 11, 16, 17, 10, 7] 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 $thetav$ 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) [5, 14] 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 [22, 3, 4]. We call these *stochastic gradient Hamiltonian dynamics* (SGHD) algorithms. SGHD 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 [22], using friction to prevent accumulation of errors in the Hamiltonian [3], and using a thermostat to control the temperature of the Hamiltonian [4]. 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 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 8) 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 do 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 ϵ 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 [8]) can squash the gains brought by the Hamiltonian dynamics. Fortunately, we can use existing stochastic approximation algorithms [18, 19] 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) [18] is described in Section 4

A further innovation of this paper is the use of common random numbers (CRN) to improve the efficiency of the Hamiltonian dynamics. The idea behind CRNs 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 [9] (and is analogous to using the same mini-batch in stochastic gradient methods). We extend and simplify this approach by including the random seeds ω 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 this doing this produces is a valid MCMC procedure. This approach is not exclusive to HABC; we apply successfully to ABC-MCMC as well.

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 CRNs 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 ([KEEP?]). The simulator is treated as 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 ϵ and treated as likelihoods. This is the equivalent to putting an ϵ-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) [11, 23, 17] 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,

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

If the simulations are part of the Markov chain, the algorithm corresponds to the pseudo-marginal (PM) sampler [1], otherwise it is a marginal sampler [11, 17]. 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. kernel density estimate [21] or a Gaussian model [25]. 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 [25]. 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 ϵ-kernel, we can convolve the two densities

$$\pi_{\boldsymbol{\epsilon}}(\mathbf{y}|\boldsymbol{\theta}) = \int \mathcal{N}(\mathbf{y}|\mathbf{x}, \epsilon^2)\mathcal{N}(\mathbf{x}|\mu_{\boldsymbol{\theta}}, \sigma^2_{\boldsymbol{\theta}})d\mathbf{x} \quad (5)$$

$$= \mathcal{N}(\mathbf{y}|\mu_{\boldsymbol{\theta}}, \sigma^2_{\boldsymbol{\theta}} + \epsilon^2) \quad (6)$$

Of particular concern to this paper is the behavior of the log-likelihoods for different values of ϵ. In the ϵ-kernel case, the log-likelihood is very sensitive to small values of ϵ:

$$\log \pi_{\boldsymbol{\epsilon}}(\mathbf{y}|\boldsymbol{\theta}) = \log \sum_s \mathcal{N}(\mathbf{y}|\mathbf{x}^{(s)}, \epsilon^2) \quad (7)$$

$$= \log \mathcal{N}(\mathbf{y}|\mathbf{x}^{(s)}, \epsilon^2) + \log(1+H) \quad (8)$$

$$\approx -\log \epsilon - \frac{1}{2\epsilon^2}(\mathbf{y} - \mathbf{x}^{(m)})^2 \quad (9)$$

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 ϵ. On the other hand, by using a model of the simulation at $\boldsymbol{\theta}$

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

For the SL model, ϵ 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 ϵ 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 existing 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 batches of the full dataset. After an overview of HMC, we will briefly describe stochastic gradient Hamiltonian dynamics (SGHDs), starting with using Langevin dynamics [22], then HMC with friction [3], and finally HMC with thermostats [4]. We will then make the connection between SGHDs 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 [5, 14] 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 η. If η 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 $K(\boldsymbol{\rho}) = \boldsymbol{\rho}^T M^{-1}\boldsymbol{\rho}/2$ (M is a diagonal matrix of masses which for presentation are set to 1). The potential energy is defined by 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 batch estimate, as is done for the following *stochastic gradient Hamiltonian dynamics* (SGHDs) 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 η (SGLD), or adding friction terms A (SGHMC), or using a thermostat ξ (SGNHT), the expensive MH correction step can be avoided and values of $\boldsymbol{\theta}$ from the Hamiltonian dynamics can be used as samples from the posterior.

SGHDs replace the full potential energy and its gradient with a batch approximation:

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

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

where n is the batch size, and h_i are indices chosen randomly without replacement from $[1, N]$ (i.e. it defined a random batch).

Stochastic gradient Langevin dynamics [22] performs one full leap-frog step of HMC. Starting with a half step for the momentum, the update for $\boldsymbol{\theta}$ is

$$\boldsymbol{\rho}_t \sim \mathcal{N}(0, \boldsymbol{I}_p) \quad (15)$$

$$\boldsymbol{\rho}_{t+\frac{1}{2}} = \boldsymbol{\rho}_t - \eta \nabla \hat{U}(\boldsymbol{\theta}_t)/2 \quad (16)$$

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \eta \boldsymbol{\rho}_{t+\frac{1}{2}} \quad (17)$$

Tt is not necessary to include $\boldsymbol{\rho}$ in the updates since there is only one step:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \eta \mathcal{N}(0, \boldsymbol{I}_p) - \eta^2 \nabla \hat{U}(\boldsymbol{\theta}_t)/2 \quad (18)$$

One of the potential drawbacks of SGLD is that the momentum term is *refreshed* for every update of the parameters, 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.

Stochastic Gradient HMC (SGHMC) [3] avoids $\boldsymbol{\rho}$ refreshment altogether. By applying HMC directly using the stochastic approximation $\hat{U}$ and $\nabla \hat{U}$, which the authors call *naive SGHMC*, the variance of the gradient will introduce errors that left unaddressed will result in sampling from the incorrect target distribution. Under the assumption that $\nabla \hat{U}(\boldsymbol{\theta}) = \nabla U(\boldsymbol{\theta}) + \mathcal{N}(\mathbf{0}, \boldsymbol{V}_{\boldsymbol{\theta}})$, where $\boldsymbol{V}_{\boldsymbol{\theta}}$ is the covariance of the gradient approximation, and updates $\boldsymbol{\rho}_{t+1} = \boldsymbol{\rho}_t + \Delta \boldsymbol{\rho}_t$ and $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \eta \boldsymbol{\rho}_{t+1}$, the change in momenta $\Delta \boldsymbol{\rho}$ from one full step is

$$-\eta \left(\nabla U(\boldsymbol{\theta}) + \mathcal{N}(\mathbf{0}, \boldsymbol{V}_{\boldsymbol{\theta}}) \right) = -\eta \nabla U(\boldsymbol{\theta}) + \mathcal{N}\left(\mathbf{0}, \eta^2 \boldsymbol{V}_{\boldsymbol{\theta}}\right) \quad (19)$$

By adding a friction term $\boldsymbol{B}$ to $\Delta \boldsymbol{\rho}$ proportional to $\boldsymbol{V}_{\boldsymbol{\theta}}$, the correction step can be avoided

$$\Delta \boldsymbol{\rho} = -\eta \boldsymbol{B} \boldsymbol{\rho}_t - \eta \nabla U(\boldsymbol{\theta}_t) + \mathcal{N}(\mathbf{0}, 2\eta \boldsymbol{B}) \quad (20)$$

where $\boldsymbol{B} = \frac{1}{2} \eta \boldsymbol{V}_{\boldsymbol{\theta}_t}$. In practice, since we can only estimate $\boldsymbol{B}$ by some $\hat{\boldsymbol{B}}$ and can only compute $\hat{U}$, a user defined friction term $\boldsymbol{C}$ is used (with $\boldsymbol{C} - \hat{\boldsymbol{B}}$ is semi-positive definite). Thus the updates used for $\Delta \boldsymbol{\rho}$ for SGHMC:

$$-\eta \boldsymbol{C} \boldsymbol{\rho}_t - \eta \nabla \hat{U}(\boldsymbol{\theta}_t) + \mathcal{N}\left(\mathbf{0}, 2\eta(\boldsymbol{C} - \hat{\boldsymbol{B}})\right) \quad (21)$$

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 (SGT) [4] addresses the difficulty of estimating $\hat{\boldsymbol{B}}$ by introducing a scalar variable ξ who's addition to the Hamiltonian dynamics maintains the temperature of the system constant, i.e. it acts as a (Nose-Hoovier) thermostat [**?**]. The update equations remain simple: initialize $\xi = \boldsymbol{C}$ (or c), then for $t = 1 \ldots$

$$\boldsymbol{\rho}_{t+1} = \rho_t - \eta \xi_t \boldsymbol{\rho} - \eta \nabla \hat{U}(\boldsymbol{\theta}_t) + \mathcal{N}(\mathbf{0}, 2\eta_t \boldsymbol{C}) \quad (22)$$

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \eta \boldsymbol{\rho}_{t+1} \quad (23)$$

$$\xi_{t+1} = \xi_t + \eta \left(\boldsymbol{\rho}_{t+1}^T \boldsymbol{\rho}_{t+1}/p - 1 \right) \quad (24)$$

In summary, the hyperparameters required for these algorithms are η and $\boldsymbol{C}$ (for SGHMC and SGNHT only), and in practice, some way of estimating $\hat{\boldsymbol{V}}$ for SGHMC.

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 [22, 3, 4] and focus on the details of how stochastic gradients are computed in the likelihood-free setting.

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 if 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-ϵ 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 η, eliminating the usefulness of HMC for ABC. Under the deterministic rep-

Algorithm 1 ∇U FDSA-ABC

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

resentation of $\mathbf{x}^{(s)}$, we can write the loglikelihoods

$$\mathcal{L}_{\boldsymbol{\epsilon}}(\boldsymbol{\theta}) \propto \log \sum_s \mathcal{N}(\mathbf{y}|f(\boldsymbol{\theta}, \omega_s), \epsilon^2) \quad (25)$$

$$\approx -\log \epsilon - \frac{1}{2\epsilon^2}(\mathbf{y} - f(\boldsymbol{\theta}, \omega_m))^2 \quad (26)$$

In the second line we have again assumes ϵ is very small and m is the index of the random seed producing the closest simulation to $\mathbf{y}$. For a finite difference approximation, $\partial \mathcal{L}_{\boldsymbol{\epsilon}}(\boldsymbol{\theta})/\partial \boldsymbol{\theta}$ is

$$\frac{1}{4d_{\boldsymbol{\theta}}\epsilon^2}\left((\mathbf{y} - f(\boldsymbol{\theta} - d_{\boldsymbol{\theta}}, \omega_m^-))^2 - (\mathbf{y} - f(\boldsymbol{\theta} + d_{\boldsymbol{\theta}}, \omega_m^+))^2\right) \quad (27)$$

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

$$\mu_{\boldsymbol{\theta}} = \frac{1}{S}\sum_s f(\boldsymbol{\theta}, \omega_s) \qquad \sigma_{\boldsymbol{\theta}}^s = \frac{1}{S-1}\sum_s (\mu_{\boldsymbol{\theta}} - f(\boldsymbol{\theta}, \omega_s))^2 \quad (28)$$

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

$$-\frac{1}{2}\log\left(\frac{\sigma_{\theta^+}^2 + \epsilon^2}{\sigma_{\theta^-}^2 + \epsilon^2}\right) - \frac{(\mathbf{y} - \mu_{\boldsymbol{\theta}^+})^2}{2(\sigma_{\boldsymbol{\theta}^+}^2 + \epsilon^2)} + \frac{(\mathbf{y} - \mu_{\boldsymbol{\theta}^-})^2}{2(\sigma_{\boldsymbol{\theta}^-}^2 + \epsilon^2)} \quad (29)$$

4.3 From Finite Differences to Simultaneous Perturbations

Algorithm 1 shows the *finite difference stochastic approximation* (FDSA) [8] to $\nabla U(\boldsymbol{\theta})$ as a function of random seeds $\boldsymbol{\omega}$. Note we have deliberately shown the deterministic simulations (f) outside of $\mathcal{L}_{\boldsymbol{\epsilon}}$ to emphasize its dependence on $\mathbf{x}$. The number of simulations required for FDSA is $2SD$, which may be acceptable for some small ABC problems. Our goal is to scale ABC to high-dimensions and for that we need an alternative stochastic approximation of $\nabla U(\boldsymbol{\theta})$.

In the gradient-free setting, Spall [18, 19] 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 $D = |\boldsymbol{\theta}|$ where entry $\delta_d \sim$ Bernouilli(1/2) (actually, values ± 1, not 0/1). Then L is evaluated at $\boldsymbol{\theta} + d_{\boldsymbol{\theta}}\boldsymbol{\Delta}$ and $\boldsymbol{\theta} - d_{\boldsymbol{\theta}}\boldsymbol{\Delta}$, giving the gradient approximation $\hat{\boldsymbol{g}}(\boldsymbol{\theta}) \approx \partial L(\boldsymbol{\theta})/\partial \boldsymbol{\theta}$:

Algorithm 2 ∇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
    $\boldsymbol{\Delta} \sim$ Bernouilli $(1/2, |\boldsymbol{\theta}|)$
    for $s = 1 : |\boldsymbol{\omega}|$ do
        $\mathbf{x}_+^{(s)} \leftarrow f(\boldsymbol{\theta} + d_{\boldsymbol{\theta}}\boldsymbol{\Delta}, \omega_s)$
        $\mathbf{x}_-^{(s)} \leftarrow f(\boldsymbol{\theta} - d_{\boldsymbol{\theta}}\boldsymbol{\Delta}, \omega_s)$
    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 $-\hat{\boldsymbol{g}}$
```

$$\hat{\boldsymbol{g}}(\boldsymbol{\theta}) = \frac{L(\boldsymbol{\theta} + d_{\boldsymbol{\theta}}\boldsymbol{\Delta}) - L(\boldsymbol{\theta} - d_{\boldsymbol{\theta}}\boldsymbol{\Delta})}{2d_{\boldsymbol{\theta}}}\begin{bmatrix} 1/\Delta_1 \\ 1/\Delta_2 \\ \vdots \\ 1/\Delta_D \end{bmatrix} \quad (30)$$

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 2 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 [19] (we use what Spall calls 2SPSA) and an algorithm for estimating the Hessian based on the same principal as SPSA [20]. The one-sided version is attractive computationally, but for HABC, the updates for $\boldsymbol{\theta}$ require simulating two-sides anyway, so using 2SPSA makes more sense. SPSA has also been used within a procedure for maximum-likelihood estimation for hidden Markov models using ABC [6].

4.4 Sticky Random Numbers

An obvious use of common random numbers, as previously demonstrated for SPSA [9], is within the gradient approximation so that the noise from the simulator has less of an influence that $d_{\boldsymbol{\theta}}$. In addition to this use, we have found that using the same random seeds over multiple time-steps improves the performance of HABC (TBD). This is very similar to using dependent random streams to drive MCMC [13, 15], the main difference we believe is that we are using

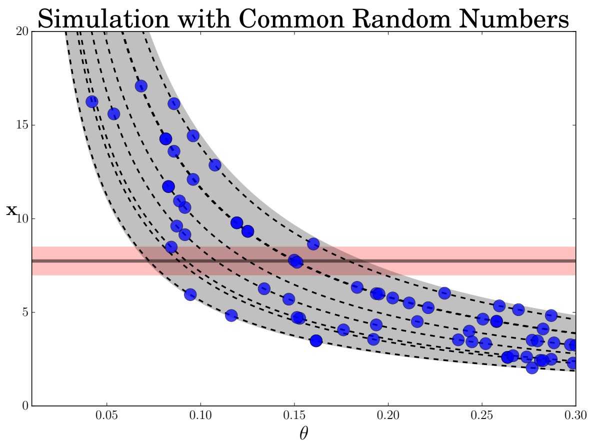

Figure 1: A view of a simulator in terms of common random numbers. The horizontal line represents $\mathbf{y}$ and red shading $\pm 2\epsilon$. The shaded curved region represents 2σ of $\pi(\mathbf{x}|\boldsymbol{\theta})$. The dashed lines are $f(\boldsymbol{\theta}, \omega_s)$ for several values of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

the Hamiltonian dynamics to drive proposal in $\boldsymbol{\theta}$ and using $\boldsymbol{\omega}$ to give persistent simulations, reducing the variance from the unknown simulator noise function.

Our updates for $\boldsymbol{\omega}$ are simple. At each time step t, each seed is refreshed to a new random value with probability γ, i.e. $\omega_s \leftarrow \omega_s$ with probability γ, and $\omega_s \leftarrow \mathcal{U}(1, W)$, where $W = 10^8$ in our algorithms. By observing the simulation at different $\boldsymbol{\theta}$ but using same seeds, we can eliminate much of the simulator noise from the Markov chain. By treating $\boldsymbol{\omega}$ as part of the state of the Markov chain, we can smoothly move along noiseless outputs for several iterations and then propose new seeds. Though proposing new seeds produces a valid Metropolis-Hastings algorithm (keeping $\boldsymbol{\theta}$ fixed and changing $\boldsymbol{\omega}$), we found that using *sticky* random seeds works well. These seeds can be precomputed before the chain is run and can be seen as apart from the state of the Markov chain. [CONFUSING]

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 ω_s; i.e. for several ω_s we vary θ (dashed lines are function $f(\theta, \omega_s)$) and randomly reveal data (blue circles).

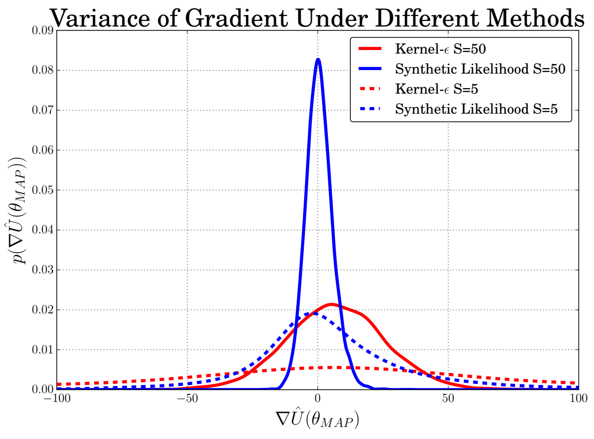

Figure 2: Variance of gradient estimation using kernel-ϵ and SL for different values of $S \in \{5, 50\}$ and fixed $\epsilon = 0.37$ (the same used in the other results). For kernel-ϵ, $\boldsymbol{V}(\boldsymbol{\theta}_{\text{MAP}}) = 6.7 \pm 131.5$ and $\boldsymbol{V}(\boldsymbol{\theta}_{\text{MAP}}) = 6.9 \pm 18.9$ (for $S = 5$ and $S = 50$, respectively). For SL,$\boldsymbol{V}(\boldsymbol{\theta}_{\text{MAP}}) = 6.35 \pm 42.4$ and $\boldsymbol{V}(\boldsymbol{\theta}_{\text{MAP}}) = 0.62 \pm 5.2$ (for $S = 5$ and $S = 50$, respectively). The "true" ABC gradient at $\boldsymbol{\theta}_{\text{MAP}}$ is 0.26. The "true" ABC loglikelihood is based on a Gaussian approximation (smoothed by ϵ), with mean and variance set by moment matching with the true Gamma distribution (the sum of N exponentials). Not only does kernel-ϵ keep a high variance for even large S, it seems to have a bias in its gradient estimate. [look into this?]

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 $\nabla\hat{U}(\boldsymbol{\theta})$

At θ_{MAP} we estimated the gradient $\nabla\hat{U}(\theta)$ using finite differences for both the kernel-$epsilon$ and synthetic-likelihood representation of $\pi_{eps}(y|x)$. For $S = 5$ and $S = 50$, $10K$ repetitions of computing the gradients was performed; there histograms are shown in Figure 4. At $S = 5$ both methods reveal a small bias (values X and Y for kernel-ϵ and SL, respectively). At $S = 50$, the bias for kernel-ϵ is much smaller, but its variance remains high; for SL, the bias remains unchanged but the variance has decreased significantly (values X pm S and Y pm T). The bias in the SL representation comes from the Gaussian approximation $\pi(y|\theta)$ which we can compute analytically using the CLT as $\text{N}(1/\theta, 1/N\theta^2 + \epsilon^2)$. (We believe this bias is due to the normal approximation assuming a symmetric mode as the bias goes to zero at the mean of the true posterior). Note that this bias is directional signal only appears nears the mode. For the $S = 5$ setting in our experiment both approaches contain similar bias. [REWRITE END]

5.2 Posterior Inference using HABC

In Figures 3 we show the posterior trace plots for ABC-MCMC, SGLD, SGHMC, and SGNHT versions of HABC using SL gradient estimates ($S = 5$). The behavior of the

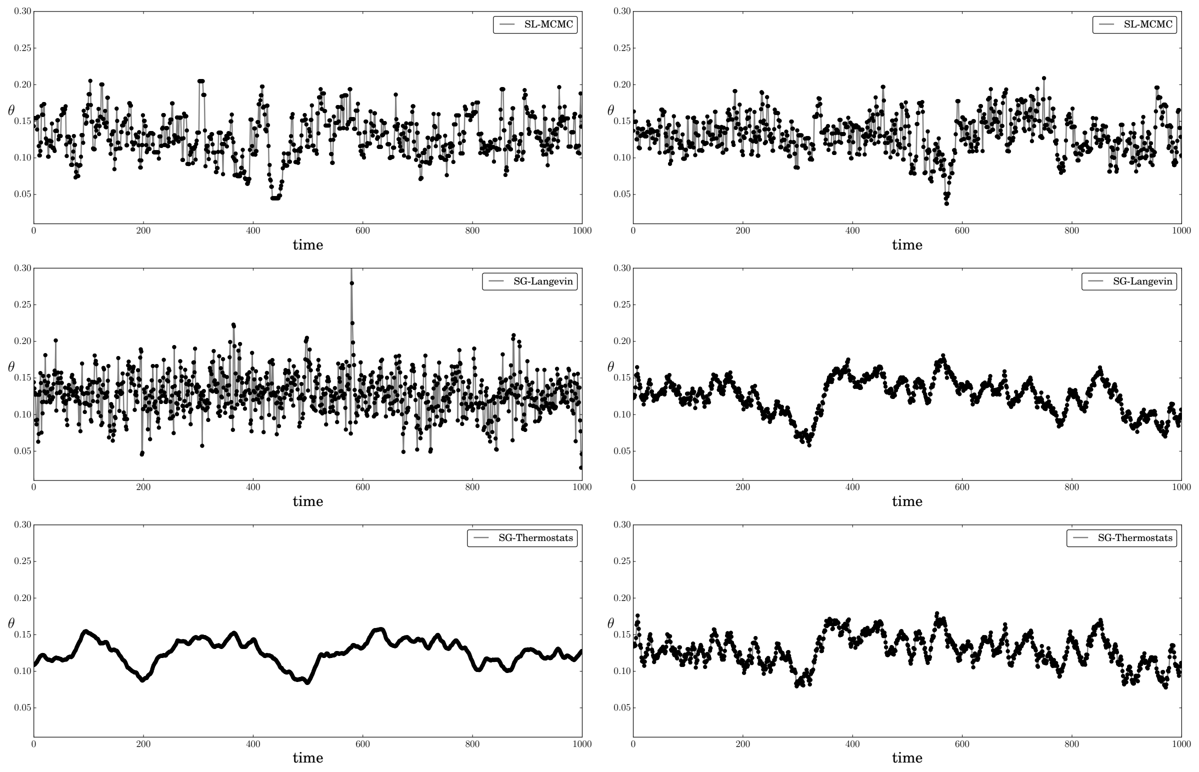

Figure 4: No sticky numbers Explanation thermostats: with no sticky, need to lower eta, so it shows smoother trajectory. With sticky, the gradients are more consistent, allowing for a larger eta and therfore more noise injection.

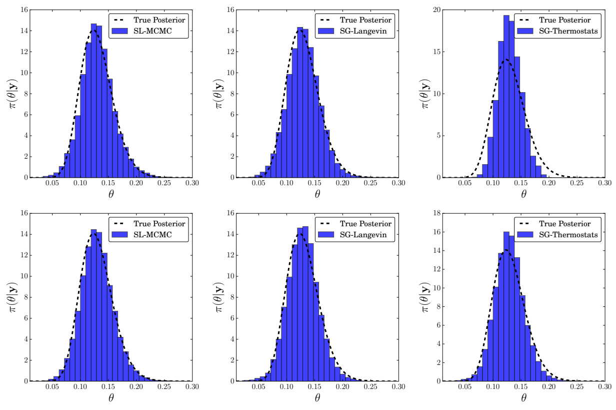

Figure 3: No sticky numbers

Table 1: Posterior inference for Exponential problem

Algo	1000	10000	50000
SL-ABC	0.099	0.047	0.045
SGLD	0.082	0.049	0.048
SGNHT	0.271	0.162	0.161
SGNHT*	0.144	0.056	0.075

trajectories is quite different, though they all produce the same correct posterior estimates (Figure 2). In Table **??** we report the *total variational distance* between the true posterior and the ABC posteriors after different numbers of samples (repeated 5 times).

5.3 Sticky Random Numbers

Finally we compare posterior estimates using sticky CRNs. Using a flip rate of 1%, i.e. each ω_s has 1% chance of changing randomly at each time-step. In Figure **??** we show the traces plots of $< x^{(s)>}$ for the different methods. While the Hamiltonian dynamics provide persistence in $\boldsymbol{\theta}$-space, the sticky random numbers do the same for the ABC statistics. The Table **??** we report the same *tvd* error using sticky numbers. [DISCUSS difference with no-sticky]. The potential benefit of using persistent random seeds is shown for all methods, including ABC-MCMC.

6 Experiments

We present experimental results comparing HABC with standard ABC-MCMC for two challenging simulators. The first is low-dimensional ($D = 6$), but exhibits chaotic behavior. For the second problem we apply the techniques of HABC to a Bayesian image classifier and autoencoder. The benefit of this problem is we can compare directly with the SGHD algorithms using the analytic gradients. This problem is has very high-dimension ($D > 15000$), yet we are able to successfully apply SPSA-based gradients to HABC.

6.1 Blowfly

For these experiments, a simulator of adult sheep blowfly populations [25] is used with statistics set to those from [12]. 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 τ is an integer. In total, there are $D = 6$ parameters $\theta = \{\log P, \log\delta, \log N_0, \log\sigma_d, \log\sigma_p, \tau\}$. As [12] we place broad log-normal priors over $\theta_{1\ldots5}$ and a Poisson prior over τ. 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 [12] 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.

Test for more accurate gradients

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 [12]) and $S = 10$ for all experiments. We use SPSA with $R = 1$ using SL log-likelihoods for all HABC gradient estimates. Note that this requires $2S$ forward simulations per time-step, the same as marginal ABC and twice pseudo-marginal ABC.

Figure **??** show the posterior distributions for $\log P, \log\delta$ and the posterior predictive distributions of [FILL IN]. Traces of $\boldsymbol{\theta}$ are shown for SGHMC and SGNHT in Figure **??**. Finally, we compare the convergence to $\mathbf{y}$ using the online posterior predictive distribution in Figure **??**. By using stick random numbers, the convergence is [FILL IN].

6.2 Bayesian Autoencoders

We perform Bayesian inference on an autoencoder neural network using MNIST images as observations. We apply two inference algorithms: SGNHT and HABC. For SGNHT, we use the known gradient and for HABC we apply 2SPSA. After 1000 iterations we collect parameter vectors every 100 iterations for a total of 100 sets. (Do we use SGHMC to estimate Bhat in first 1000 iterations?). Use same batches at each iteration for SGNHT and HABC?

results: convergence of training error (batches), convergence in test error using parameters collected (record nbr of forward or backward passes required), show mean and variance of filters (how?), compare with using the last parameter sample only (stochastic optimization).

7 DISCUSSION

8 CONCLUSION

- New set of ABC algorithms for efficient exploration of posterior distribution.
- Key: opens the door to high-dimensional ABC inference.
- For ABC, the gradients we compute are stochastic because we must approximate $\pi(\mathbf{x}|\boldsymbol{\theta})$ with forward simulations. We can make the approximation arbitrarily good by running more simulations, but this is often unnecessary for sampling, because we can make use of the uncertainty in our current approximation to decide if our sample is correct or not (cite UAI2014?).
- Unbiased gradients with low noise. SL bias might be intolerable; use KDE to get stable unbiased gradients?

A further complication, which we do not address on this paper, is the forward simulation might be, and for many interesting problems is, expensive. We would therefore prefer to resort to some sort of surrogate modeling [12, 24**?**] to avoid simulations whenever possible. We address this scenario in Section **??**. Shown for relatively fast simulators. Extensions to surrogates with GPS (Meeds2014GpsUai, rasmussen:2003, wilkinson:2014) can use derivative of GP for likelihood as GP of gradient. These could be matched by occasional gradient estimates (Osborne).

[1] Equation 1 in Section 1.2.3 of the supplementary information in [25].

References

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

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

[3] Chen, Tianqi, Fox, Emily B, and Guestrin, Carlos. Stochastic gradient hamiltonian monte carlo. 2014.

[4] 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.

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

[6] 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.

[7] Fan, Yanan, Nott, David J, and Sisson, Scott A. Approximate bayesian computation via regression density estimation. *Stat*, 2013.

[8] 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.

[9] 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.

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

[11] 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.

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

[13] Murray, Iain and Elliott, Lloyd T. Driving markov chain monte carlo with a dependent random stream. *arXiv:1204.3187*, 2012.

[14] Neal, Radford M. Mcmc using hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*, 2, 2011.

[15] 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.

[16] Sisson, SA, Fan, Y, and Tanaka, Mark M. Sequential monte carlo without likelihoods. *Proceedings of the National Academy of Sciences*, 104(6):1760, 2007.

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

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

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

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

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

[22] 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.

[23] 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.

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

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