# Adaptive Tuning of Hamiltonian Monte Carlo Within Sequential Monte Carlo

Alexander Buchholz, Nicolas Chopin, Pierre Jacob

Bayesian Analysis (2021) **16**, Number 3, pp. 745–771

# Adaptive Tuning of Hamiltonian Monte Carlo Within Sequential Monte Carlo

Alexander Buchholz*, Nicolas Chopin†, and Pierre E. Jacob‡

**Abstract.** Sequential Monte Carlo (SMC) samplers are an alternative to MCMC for Bayesian computation. However, their performance depends strongly on the Markov kernels used to rejuvenate particles. We discuss how to calibrate automatically (using the current particles) Hamiltonian Monte Carlo kernels within SMC. To do so, we build upon the adaptive SMC approach of Fearnhead and Taylor (2013), and we also suggest alternative methods. We illustrate the advantages of using HMC kernels within an SMC sampler via an extensive numerical study.

**Keywords:** Sequential Monte Carlo, Hamiltonian Monte Carlo.

**MSC2020 subject classifications:** Primary 62F15; secondary 65C05.

## 1 Introduction

Sequential Monte Carlo (SMC) samplers (Neal, 2001; Chopin, 2002; Del Moral et al., 2006) approximate a target distribution $\pi$ by sampling particles from an initial distribution $\pi_0$, and moving them through a sequence of distributions $\pi_t$ which ends at $\pi_T = \pi$. In Bayesian computation this approach has several advantages over Markov chain Monte Carlo (MCMC). In particular, it enables the estimation of normalizing constants and can thus be used for model choice (Zhou et al., 2016). Moreover, particles can be propagated mostly in parallel (Murray et al., 2016). Finally, SMC samplers are more robust to multimodality (Schweizer, 2012b; Jasra et al., 2015).

SMC samplers iterate over a sequence of resampling, propagation and reweighting steps. The propagation of the particles commonly relies on MCMC kernels, which depend on some tuning parameters. Choosing these parameters in a sensible manner is challenging and is of interest both from a theoretical and practical point of view; see Fearnhead and Taylor (2013); Schäfer and Chopin (2013); Beskos et al. (2016).

One type of MCMC kernels that has raised attention recently is Hamiltonian Monte Carlo (HMC), originally developed in physics (Duane et al., 1987) and used in statistics since Neal (1993). It has become a standard tool for sampling distributions with continuously differentiable densities (Neal, 2011). The main appeal of HMC is its better mixing (compared to, say, random walk Metropolis) in high-dimensional problems (Beskos et al., 2013; Mangoubi and Smith, 2017).

This paper compares methods for tuning HMC kernels within SMC. A few previous works have considered the use of HMC kernels within SMC (Gunawan et al., 2018;

*MRC Biostatistics Unit, University of Cambridge, UK, ab2603@cantab.ac.uk
†ENSAE-CREST, Institut Polytechnique, Paris, France
‡Department of Statistics, Harvard University, USA

 https://doi.org/10.1214/20-BA1222

Burda and Daviet, 2018; Daviet, 2018; Kostov, 2016) without focusing on tuning. Tuning MCMC kernels within SMC has a significant impact on performance, and particularly so for HMC kernels. Calibration of HMC kernels is indeed recognised as a challenging problem in the MCMC literature (e.g. Mohamed et al., 2013; Beskos et al., 2013; Betancourt et al., 2014; Betancourt, 2016; Hoffman and Gelman, 2014). The advantage of tuning Markov kernels within SMC is that a cloud of particles is available to inform on the shape and scale of the current target distribution.

We base our approach on the work of Fearnhead and Taylor (2013), which concerned the tuning of generic MCMC kernels within SMC samplers, and on existing approaches to tuning HMC. We apply the proposed SMC sampler with HMC kernels to five examples; three toy examples, a binary Bayesian regression of dimension up to 95 and a log Gaussian Cox model of dimension up to 16,384. Our numerical study illustrates the performance of SMC samplers for inference and model choice in high dimensions, and the benefits brought by HMC relative to random walk and Langevin kernels. We also investigate the effect of the tempering ladder and of the number of move steps. The paper is organized as follows. Section 2 reviews SMC samplers and HMC kernels. Section 3 discusses adaptive tuning procedures for SMC. Section 4 provides numerical experiments, and Section 5 discusses the results.

## 2 Background

We will focus on target distributions that are posterior distributions and we will use the associated terminology. We consider the problem of calculating expectations of a test function $\varphi : \mathbb{R}^d \mapsto \mathbb{R}$ with respect to a posterior distribution defined as $\pi(x) = p(x)l(y|x)/Z$. The random variable $x$ with density $\pi(\cdot)$ is defined on the space $(\mathbb{R}^d, \mathcal{B}(\mathbb{R}^d))$, where $\mathcal{B}(\mathbb{R}^d)$ denotes the Borel sets of $\mathbb{R}^d$. Here $p(x)$ denotes the prior distribution, $l(y|x)$ is the likelihood of the observed data $y$ given the parameter $x \in \mathbb{R}^d$, and $Z = \int_{\mathbb{R}^d} l(y|x)p(x)dx$ denotes the normalizing constant, also called marginal likelihood or evidence. We next describe Sequential Monte Carlo (SMC) and Hamiltonian Monte Carlo (HMC). These are building blocks for the adaptive algorithms discussed in this paper.

### 2.1 Sequential Monte Carlo samplers

Sequential Monte Carlo (SMC) approaches the problem of sampling from $\pi$ by introducing a sequence of intermediate distributions $\pi_0, \cdots, \pi_T$ defined on the common measurable target space $(\mathbb{R}^d, \mathcal{B}(\mathbb{R}^d))$, such that $\pi_0$ is easy to sample from, and $\pi_T = \pi$.

We construct intermediate distributions using tempering, that is $\pi_t(x) \propto p(x)l(y|x)^{\lambda_t}$, where the sequence of exponents $\lambda_t$ is such that $0 = \lambda_0 < \cdots < \lambda_t < \cdots < \lambda_T = 1$. These exponents may be automatically selected during the run of a SMC sampler, as described later. Other choices of sequences of distributions are possible (see e.g. Chopin, 2002; Del Moral et al., 2006; Chopin et al., 2013). We assume throughout the article that the prior distribution $p(x)$ is a proper probability distribution.

### SMC samplers with MCMC moves on tempered posteriors

We denote by $\gamma_t(x) = p(x)l(y|x)^{\lambda_t}$ the unnormalized density associated with $\pi_t(x)$, and by $Z_t$ the normalizing constant: $Z_t = \int \gamma_t(x)dx$. One way of constructing SMC samplers is as follows. Suppose that at time $t-1$ an equally weighted particle approximation $\{\tilde{x}_{t-1}^i\}_{i\in 1:N}$ of $\pi_{t-1}$ is available, with possible duplicates among the particles. This cloud of particles is then moved with a Markov kernel $\mathcal{K}_t$, that leaves the distribution $\pi_{t-1}$ invariant: for each $i$, $x_t^i \sim \mathcal{K}_t(\tilde{x}_{t-1}^i, dx)$. Consequently a set of new samples $\{x_t^i\}_{i\in 1:N}$ is obtained. These particles are then weighted: particle $x_t^i$ is assigned an importance weight $w_t^i = \gamma_t(x_t^i)/\gamma_{t-1}(x_t^i)$, so that the next distribution $\pi_t$ is approximated by the set $\{x_t^i, w_t^i\}_{i\in 1:N}$. After resampling particles according to their weights, one obtains an equally weighted set $\{\tilde{x}_t^i\}_{i\in 1:N}$ and the procedure is repeated for the next target distribution $\pi_{t+1}$. The procedure is described in Algorithm 1.

**Algorithm 1:** SMC sampler with MCMC moves on tempered posteriors.

**Input**: Number of particles $N$, distributions $\pi_t(x) \propto p(x)l(y|x)^{\lambda_t}$, rule for constructing Markov kernels $\mathcal{K}_t^h$ that are $\pi_{t-1}$ invariant.
**Result**: Set of weighted samples and normalizing constant estimates.
**Initialization**: t = 1, $\lambda_0 = 0$;
**Iteration**:

1 **while** $\lambda_{t-1} < 1$ **do**
2  **if** $t = 1$ **then**
3  **foreach** $i \in 1{:}N$ **do**
4  Sample $x_1^i \sim \pi_0$;
5  **else**
6  Tune Markov kernel parameters $h$ using particles; see Algorithm 5 or 6;
7  **foreach** $i \in 1{:}N$ **do**
8  Move particle $x_t^i \sim \mathcal{K}_t^h(\tilde{x}_{t-1}^i, dx)$;
9  (Move step can be iterated, see Algorithm 3);
10  Choose next exponent $\lambda_t \in (\lambda_{t-1}, 1]$ based on particles; see Algorithm 2;
11  **foreach** $i \in 1{:}N$ **do**
12  Weight particle $w_t^i = \frac{\gamma_t(x_t^i)}{\gamma_{t-1}(x_t^i)}$;
13  Calculate $\widehat{Z_t/Z_{t-1}} = N^{-1}\sum_{i=1}^N w_t^i$;
14  Resample particles $\{x_t^i, w_t^i\}_{i\in 1:N}$ to obtain $\{\tilde{x}_t^i\}_{i\in 1:N}$;
15  Set $t = t + 1$;

Upon completion, the algorithm returns weighted samples $\{x_t^i, w_t^i\}_{i\in 1:N}$, which may be used to estimate expectations with respect to the target distributions, in the sense that weighted averages of the form $\sum_{i=1}^N w_t^i\varphi(x_t^i)/\sum_{i=1}^N w_t^i$ converge to $\mathbb{E}_{\pi_t}[\varphi(x)]$ as $N \to +\infty$, in probability. The algorithm also returns estimates of the ratios $Z_t/Z_{t-1}$, and thus of $Z_T/Z_0$; see line 13 in Algorithm 1. One may also use the path sampling identity to derive an alternate estimate of $Z_t/Z_{t-1}$ (Zhou et al., 2016).

The kernels $\mathcal{K}_t$ may be chosen as Metropolis–Hastings (MH) kernels (see e.g. Chopin, 2002; Jasra et al., 2011; Sim et al., 2012; Fearnhead and Taylor, 2013; Zhou et al., 2016). More details on the choice of kernels and on optimality can be found in Del Moral et al. (2006, 2007). In general Markov kernels may depend on a set of tuning parameters $h$, and are hereafter denoted by $\mathcal{K}_t^h$.

### Tuning of the SMC sampler

Different design choices have to be made for the SMC sampler of Algorithm 1 to be operational.

**(a)** The choice of the next exponent $\lambda_t$, at line 10 of Algorithm 1, may be based on available particles; for instance on their effective sample size, as explained below.

**(b)** The number of move steps, at line 9 of Algorithm 1, may be based on the observed performance of the Markov kernels; see below.

**(c)** The tuning of the Markov kernel parameters $h$, at line 6 of Algorithm 1, may be based on the particles, which is the main difference with the standard MCMC setting. The main contribution of this paper is to investigate this tuning in the case of HMC kernels, and is described in Section 3.

**(a) Choice of the next exponent** A common approach (Jasra et al., 2011; Schäfer and Chopin, 2013) to choose adaptively intermediate distributions within SMC is to rely on the ESS (effective sample size, Kong et al., 1994). The ESS is a measure of performance for importance sampling estimates (Agapiou et al., 2017). This criterion is calculated as follows:

$$\mathrm{ESS}(\lambda_t) = \frac{\left(\sum_{i=1}^N w_t^i\right)^2}{\sum_{i=1}^N \left(w_t^i\right)^2}, \tag{1}$$

where $w_t^i = \gamma_t(x_t^i)/\gamma_{t-1}(x_t^i) = l(y|x_t^i)^{\lambda_t - \lambda_{t-1}}$ in the setting considered here. The ESS is a Monte Carlo approximation of $N/(1+\chi^2(\pi_t, \pi_{t-1}))$ where $\chi^2(\pi_t, \pi_{t-1})$ is the $\chi^2$ divergence from $\pi_{t-1}$ to $\pi_t$.

We may choose $\lambda_t$ by solving (in $\lambda$) the equation $\mathrm{ESS}(\lambda) = \alpha N$, for some user-chosen value $\alpha \in (0,1)$. The corresponding algorithm is described in Algorithm 2. The validity of adaptive SMC samplers based on an ESS criterion is studied in Beskos et al. (2016); Huggins and Roy (2018); Whiteley et al. (2016). Another approach for choosing the sequence of temperature steps is exposed in Friel and Pettitt (2008).

**(b) Number of move steps** The mixing of MCMC kernels plays a crucial role in the performance and stability of SMC samplers (see e.g. Del Moral et al., 2006; Schweizer, 2012a; Ridgway, 2016).

For any MCMC kernel targeting $\pi$, mixing can be improved by repeated application of the kernel, for a linear cost in the number of repetitions. We propose to monitor the product of componentwise first-order autocorrelations of the particles to decide how many repetitions to use. Autocorrelations are calculated w.r.t. $\{\tilde{x}_t^i\}_{i\in 1:N}$, the cloud of

**Algorithm 2:** Choice of the next exponent based on the effective sample size.

**Input**: Target value $\alpha$, likelihood $l(y|x_t^i)$ for the $N$ particles, current $\lambda_{t-1}$.
**Result**: Next temperature $\lambda_t$.

Define $\beta^i(\lambda) := l(y|x_t^i)^{\lambda-\lambda_{t-1}}$ and

$$\mathrm{ESS}(\lambda) = \frac{\left(\sum_{i=1}^N \beta^i(\lambda)\right)^2}{\sum_{i=1}^N \left(\beta^i(\lambda)\right)^2};$$

**if** $\mathrm{ESS}(1) \geq \alpha N$ **then**
  $\lambda_t = 1$
**else**
  Solve $\mathrm{ESS}(\lambda) = \alpha N$ in $\lambda \in [\lambda_{t-1}, 1]$, using bisection, assign result to $\lambda_t$.

particles after reweighting and resampling at time $t$. After $k$ move steps through the kernel $\mathcal{K}_t^h$ the cloud of particles is $\{x_{t,k}^i\}_{i\in 1:N}$. We then calculate the empirical correlation of the component-wise statistic $x_{t,k}^i(j) + x_{t,k}^i(j)^2$, where $x_{t,k}^i(j)$ denotes component $j$ of the vector $x_{t,k}^i$, using the successive states of the chain $x_{t,k}^i(j)$ and $x_{t,k-1}^i(j)$. This empirical correlation is denoted by $\hat{\rho}_k(j)$. This statistic is chosen to reflect the first two moments of the particles, but is otherwise arbitrary. We suggest to continue applying the Markov kernel until a large fraction (e.g. 90%) of the product of the first order autocorrelations drops below a threshold $\alpha' = 0.1$, for example. The resulting algorithm is described in Algorithm 3. Instead of using component-wise autocorrelations, one could draw on the recent work of Vats et al. (2015) on performance evaluation of MCMC, or on approaches based on the Stein discrepancy (Gorham and Mackey, 2015).

**Algorithm 3:** Adaptive move step based on autocorrelations.

**Input**: Particles $\{\tilde{x}_t^i\}_{i\in 1:N}$, proposal kernel $\mathcal{K}_t^h$.
**Result**: Particles after $k$ move steps $\{x_{t,k}^i\}_{i\in 1:N}$.
**Initialization**: $\{x_{t,0}^i\}_{i\in 1:N} \leftarrow \{\tilde{x}_t^i\}_{i\in 1:N}$, $k \leftarrow 0$.

**while** $\#\{j : \prod_{l=1}^k \hat{\rho}_l(j) > \alpha'\}/d \geq 10\%$ **do**
  Set $k \leftarrow k + 1$;
  Move particle $x_{t,k}^i \sim \mathcal{K}_t^h(x_{t,k-1}^i, dx)$ for all $i$;
  Calculate the correlation $\hat{\rho}_k(j)$ for all $j$;

## 2.2 Hamiltonian Monte Carlo

Hamiltonian Monte Carlo (HMC) consists in proposing moves by solving the equations of motion of a particle evolving in a potential. We follow the exposition in Neal (2011), before turning to the question of tuning.

### MCMC based on Hamiltonian dynamics

Let $\mathcal{L}(x) = \log \gamma(x)$ be the unnormalized log density of the random variable of interest $x$. We introduce an auxiliary random variable $p \in \mathbb{R}^d$ with distribution $\mathcal{N}(0, \mathbf{M})$, and hence unnormalized log density $\log f(p) = -1/2\ p^T \mathbf{M}^{-1} p$. The joint unnormalized density of $(p, x)$ is given as $\mu(p, x) = f(p)\gamma(x)$ and the negative joint log-density is denoted by

$$H(p, x) = -\log \mu(p, x) = -\mathcal{L}(x) + \frac{1}{2} p^T \mathbf{M}^{-1} p.$$

That function is called the Hamiltonian, with the first term representing potential energy at position $x$, and the second term representing kinetic energy, for the momentum $p$ and mass matrix $\mathbf{M}$. The trajectory of a particle with position $x$ and momentum $p$ can be described via the Hamilton equations,

$$\begin{cases} \frac{dx}{d\tau} = \frac{\partial H}{\partial p} = \mathbf{M}^{-1} p, \\ \frac{dp}{d\tau} = -\frac{\partial H}{\partial x} = \nabla_x \mathcal{L}(x), \end{cases}$$

where $dx/d\tau, dp/d\tau$ denote the derivatives of the position and the momentum with respect to the continuous time $\tau$. The solution of this differential equation induces a flow $\Phi_\tau$ that describes the evolution of a system with initial momentum and position $(p_0, x_0)$ such that $\Phi_\tau(p_0, x_0) = (p_\tau, x_\tau)$. The solution is (a) energy preserving, e.g. $H(p_\tau, x_\tau) = H(p_0, x_0)$; (b) volume preserving and consequently the determinant of the Jacobian of $\Phi_\tau$ equals one; (c) the flow is reversible w.r.t. time. In terms of probability distributions this means that if $(p_0, x_0) \sim \mu(p, x)$ then also $(p_\tau, x_\tau) \sim \mu(p, x)$.

In most cases an exact solution of the equation is not available and one resorts to numerical methods. One widely used integrator is the Störmer-Verlet or leapfrog integrator (Hairer et al., 2003). The leapfrog integrator iterates the steps:

$$\begin{aligned} p_{\tau+\epsilon/2} &= p_\tau + \epsilon/2 \nabla_x \mathcal{L}(x_\tau), \\ x_{\tau+\epsilon} &= x_\tau + \epsilon \mathbf{M}^{-1} p_{\tau+\epsilon/2}, \\ p_{\tau+\epsilon} &= p_{\tau+\epsilon/2} + \epsilon/2 \nabla_x \mathcal{L}(x_{\tau+\epsilon}), \end{aligned}$$

where $\epsilon$ is a step size. Thus, in order to let the system evolve from $\tau$ to $\tau + \kappa$ with $\kappa = L \times \epsilon$ we perform $L$ steps as given above. This induces a numerical flow $\widehat{\Phi}_{\epsilon,L}$ such that $\widehat{\Phi}_{\epsilon,L}(p_\tau, x_\tau) = (\hat{p}_{\tau+\kappa}, \hat{x}_{\tau+\kappa})$. In general we have $\Delta E_\kappa \neq 0$ where $\Delta E_\kappa = H(\hat{p}_{\tau+\kappa}, \hat{x}_{\tau+\kappa}) - H(p_\tau, x_\tau)$ is the variation of the Hamiltonian. The dynamics can be used to construct a Markov chain targeting $\mu$ on the joint space, with a MH step that corrects for the variation in energy, as described in Algorithm 4.

### Existing approaches to tuning HMC

The error analysis of geometric integration gives insights on choices of step sizes $\epsilon$ that yield stable trajectories. For the leapfrog integrator the error of the energy is

$$|H(\hat{p}_{\tau+\kappa}, \hat{x}_{\tau+\kappa}) - H(p_\tau, x_\tau)| \leq C_1 \epsilon^2, \tag{2}$$

**Algorithm 4:** Hamiltonian Monte Carlo algorithm.

**Input**: Gradient function $\nabla_x \mathcal{L}(\cdot)$, initial state $x_s$, energy function $\Delta E$
**Result**: Next state of the chain $(p_{s+1}, x_{s+1})$
Sample $p_s \sim \mathcal{N}(0_d, \mathbf{M})$
Apply the leapfrog integration: $(\hat{p}_{s+1}, \hat{x}_{s+1}) \leftarrow \widehat{\Phi}_{\epsilon,L}(p_s, x_s)$
Sample $u \sim \mathcal{U}[0,1]$
**if** $\log(u) \leq \Delta E_s$ **then**
    Set $x_{s+1} \leftarrow \hat{x}_{s+1}$
**else**
    Set $x_{s+1} \leftarrow x_s$

and the error of the position and momentum is

$$\left\| \widehat{\Phi}_{\epsilon,L}(\hat{p}_\tau, \hat{x}_\tau) - \Phi(p_\tau, x_\tau) \right\|_2 \leq C_2 \epsilon^2, \tag{3}$$

see Leimkuhler and Matthews (2016); Bou-Rabee and Sanz-Serna (2018) for more details. It can be shown that the constant $C_1 > 0$ in (2) stays stable over exponential long time intervals $\epsilon L \leq \exp(h_0/2\epsilon)$ for some constant $h_0$ (Hairer et al., 2006, Theorem 8.1), whereas the constant $C_2 > 0$ in the (3) typically grows with $L$. Hence, care must be taken when choosing $(\epsilon, L)$. Using the error control in (2), Neal (2011) following Creutz (1988) provides an informal reasoning motivating the scaling of $\varepsilon$ as $d^{-1/4}$, at least for targets that factorize into products of $d$ independent components. To maintain a fixed integration time $\varepsilon L$ one should then scale $L$ as $d^{1/4}$.

From a practical point of view the tuning of the HMC kernel requires the consideration of the following aspects. If $\epsilon$ is too large, the numerical integration of the HMC flow becomes unstable and results in large variations in the energy and thus a low acceptance rate, see (2). On the other hand if $\epsilon$ is too small, for a fixed number of steps $L$ the trajectories tend to be short and high autocorrelations will be observed, see Neal (2011). To counterbalance this effect a large $L$ would be needed and thus computation time would increase. If $L$ gets too large, the trajectories might also double back on themselves (Hoffman and Gelman, 2014).

From a theoretical perspective Beskos et al. (2013) and later Betancourt et al. (2014) show that the integrator step size $\epsilon$ should be chosen such that acceptance rates between 0.651 and 0.9 are obtained, when the dimension of the target space goes to infinity. This idea has been exploited in Hoffman and Gelman (2014) where stochastic approximation is used to tune $\epsilon$.

A different approach is to choose tuning parameters such that the expected squared jumping distance (ESJD) is maximized, see Pasarica and Gelman (2010), where

$$\text{ESJD} = \mathbb{E}\left[\|x_s - x_{s-1}\|_2^2\right] = 2(1-\rho_1)\,\text{Var}_\pi[x],$$

in one dimension, assuming stationarity. In this sense maximizing the ESJD of a Markov chain is equivalent to minimizing the first order autocorrelation $\rho_1$. In $d$ dimensions

maximizing the ESJD in Euclidean norm amounts to minimizing the correlation of the $d$ dimensional process in Euclidean norm. In the context of HMC this has been discussed in Mohamed et al. (2013); Hoffman and Gelman (2014). Mohamed et al. (2013) tune the HMC sampler with Bayesian optimization and vanishing adaptation, in the spirit of adaptive MCMC (Andrieu and Thoms, 2008). The ESJD is then maximized as a function of $(\epsilon, L)$. Hoffman and Gelman (2014) discuss the ESJD as a criterion for tuning $L$. At a high level the simulation of trajectories could be interrupted when the ESJD starts to decrease. However, a naive application of the idea impacts the reversibility of the Markov kernel. This problem can be solved by adjusting the acceptance mechanism as in NUTS (No U-Turn Sampler, Hoffman and Gelman, 2014), a technique employed in the probabilistic programming language Stan (Carpenter et al., 2017).

Neal (2011) suggests preliminary runs to find reasonable values of $(\epsilon, L)$ and proposes to randomize these values. The randomization avoids some pathological behavior that might occur when $(\epsilon, L)$ are poorly selected. Other approaches on identifying the optimal trajectory length are discussed in Betancourt (2016).

Another important tuning parameter is the mass matrix $\mathbf{M}$ used to sample the momentum. When the target distribution is close to a Gaussian, rescaling the target by the Cholesky decomposition of the inverse covariance matrix eliminates the correlation of the target and can improve the performance of the sampler. Equivalently, the inverse covariance matrix can be set to the mass matrix of the momentum (Neal, 2011). Recently, Girolami and Calderhead (2011) suggest using a position dependent mass matrix that takes local curvature into account.

# 3 Tuning of Hamiltonian Monte Carlo within Sequential Monte Carlo

We now discuss the tuning of the Markov kernel in line 6 of Algorithm 1. The tuning of Markov kernels within SMC samplers has the advantage that information on the intermediate distributions is available from the particles. Moreover, different kernel parameters can be assigned to different particles so that various parameters can be tested in parallel. These points were made in Fearnhead and Taylor (2013), which we will follow closely. We first describe the tuning of the mass matrix. Second, we present our adaptation of the approach of Fearnhead and Taylor (2013) to the tuning of HMC kernels, abbreviated by FT. Then we present an alternative approach based on a pretuning phase at each intermediate step, abbreviated by PR for preliminary run. Finally, we discuss the advantages and drawbacks of the two approaches.

## 3.1 Tuning of the mass matrix of the kernels

The HMC kernels depend on a mass matrix $\mathbf{M}$ used to sample momentum variables. We will exploit the information in the available particles to calibrate the mass matrix, following what was done in Chopin (2002); South et al. (2019) for the covariance of HM

proposals. Specifically we use the matrix $\mathbf{M}_t$ at iteration $t$,

$$\mathbf{M}_t = \text{diag}(\widehat{\text{Var}}_{\pi_t}[x_t])^{-1}, \tag{4}$$

where $\widehat{\text{Var}}_{\pi_t}[x_t]$ is an estimate of the covariance of $\pi_t$. The restriction to a diagonal matrix makes this approach more applicable in high dimensions. Alternatively a full covariance or precision matrix could be estimated with sparsity assumptions (e.g. Liu et al., 2016).

## 3.2 Adapting the tuning procedure of Fearnhead and Taylor (2013)

Consider line 6 during iteration $t$ of Algorithm 1, and the task of choosing parameters $h$ for the propagation kernel $\mathcal{K}_t^h$. Fearnhead and Taylor (2013) consider the ESJD criterion:

$$g_t(h) = \int \pi_{t-1}(dx_{t-1})\mathcal{K}_t^h(x_{t-1}, dx_t)\|x_{t-1} - x_t\|_M^2, \tag{5}$$

where $\|x - y\|_M^2 = (x-y)^t M^{-1}(x-y)$ stands for the Mahalanobis distance with respect to matrix $M$; in our case we set $M = \mathbf{M}_{t-1}$, see (4). Fearnhead and Taylor (2013) set $M$ to the full covariance matrix of the particles at time $t-1$, but, again, this can incur computing costs that are cubic in the dimension which may be too expensive in high-dimensional problems. By maximizing $g_t(h)$ we minimize the first-order autocorrelation of the chain. This leads to a reduced asymptotic variance of the chain and hopefully to a reduced asymptotic variance for estimates obtained from the SMC sampler.

The tuning procedure referred to as FT has the following steps:

1. Assign different values of $h_t^i$ according to their performance to the resampled particles $\tilde{x}_{t-1}^i$.
2. Propagate $x_t^i \sim \mathcal{K}_t^{h_t^i}(\tilde{x}_{t-1}^i, dx)$.
3. Evaluate the performance of $h_t^i$ based on $x_t^i, \tilde{x}_{t-1}^i$.

In Step 1, we use the following performance metric, which is a Rao-Blackwellized estimator of (5):

$$\tilde{\Lambda}(\tilde{x}_{t-1}^i, \hat{x}_t^i) = \frac{\left\|\tilde{x}_{t-1}^i - \hat{x}_t^i\right\|_M^2}{L} \times \min(1, \exp[\Delta E_t^i]). \tag{6}$$

Here $\hat{x}_t^i$ is the proposed position based on the Hamiltonian flow $\widehat{\Phi}_{\epsilon,L}(\tilde{x}_{t-1}^i, p_t^i)$ before the MH step. The acceptance rate $\min(1, \exp[\Delta E_t^i])$ of the MH step is based on the variation of the energy $\Delta E_t^i$, and serves as weight. This metric is the same as in Fearnhead and Taylor (2013), except that it is divided by $L$, in order to account for the fact that the CPU (Central Processing Unit) cost of an HMC kernel increases linearly with $L$.

The pairs $h_t^i = (\epsilon_t^i, L_t^i)$ are weighted according to the performance metric $\tilde{\Lambda}(\tilde{x}_{t-1}^i, \hat{x}_t^i)$. The next set of parameters $h_{t+1}^i$ are sampled from

$$\chi_{t+1}(h) \propto \sum_{i=1}^{N} \tilde{\Lambda}(\tilde{x}_{t-1}^i, \hat{x}_t^i) R(h; h_t^i),$$

where $R$ is a perturbation kernel. We suggest to set $R$ to

$$R(h; h_{t-1}^i) = \mathcal{TN}(\epsilon; \epsilon_{t-1}^i, 0.015^2) \otimes \left\{ \frac{1}{3}\mathbb{1}_{\{L_{t-1}^i - 1\}}(L) + \frac{1}{3}\mathbb{1}_{\{L_{t-1}^i\}}(L) + \frac{1}{3}\mathbb{1}_{\{L_{t-1}^i + 1\}}(L) \right\},$$

where $\mathcal{TN}$ denotes a normal distribution truncated to $\mathbb{R}^+$. The part in curly brackets corresponds to a discrete mixture for the variable $L$. Thus $\epsilon$ is perturbed by a small (truncated) Gaussian noise, and $L$ has an equal chance of increasing, decreasing or staying the same. (When $L = 1$, decreasing is forbidden, and the kernel is adapted to assign equal probabilities to either increasing $L$ or keeping it constant.) The variance of the Gaussian noise is set to the value used by Fearnhead and Taylor (2013). In our simulations we found that tuning was robust to the choice of this value. The procedure is described in Algorithm 5.

**Algorithm 5:** (FT) Tuning of HMC based on Fearnhead and Taylor (2013).

**Input**: Previous parameters $h_{t-1}^i$, estimator of associated utility $\tilde{\Lambda}(\tilde{x}_{t-2}^i, \hat{x}_{t-1}^i)$, $i \in 1{:}N$, perturbation kernel $R$

**Result**: Sample of $h_t^i = (\epsilon_t^i, L_t^i)$, $i \in 1{:}N$

**foreach** $i \in 1{:}N$ **do**

  Sample $h_t^i \sim \chi_t(h) \propto \sum_{i=1}^{N} \tilde{\Lambda}(\tilde{x}_{t-2}^i, \hat{x}_{t-1}^i) R(h; h_{t-1}^i)$;

## 3.3 Pretuning of the kernel at every step

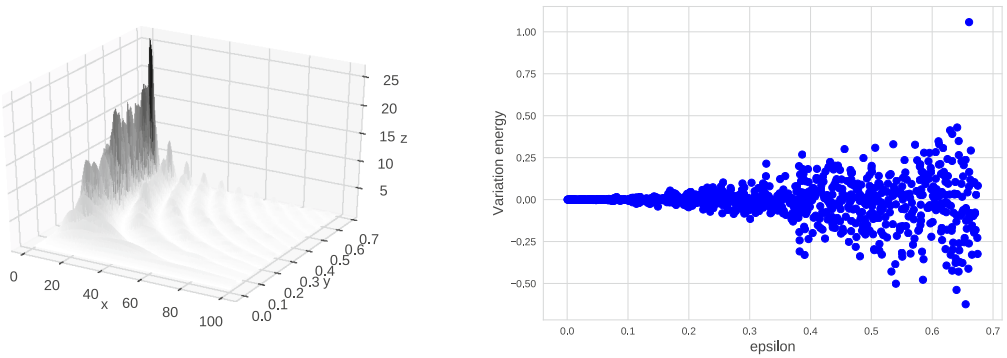
The previous tuning algorithm relies on the assumption that parameters suited for the kernel used at time $t-1$ will also achieve good performance at time $t$. We suggest the following two-stage procedure as an alternative:

1. Apply an HMC step targeting to $\pi_{t-1}$ to the $N$ current particles. For each particle the value of $(\epsilon, L)$ is chosen randomly from a certain uniform distribution (see next section). Then construct a new distribution for $(\epsilon, L)$ based on the observed performance (Section 3.3). The HMC trajectories are then discarded.

2. Apply again an HMC step to the $N$ current particles, this time with $(L, \epsilon)$ generated from the distribution constructed in the previous step.

### Range of values for $\epsilon$

In the first stage $\epsilon$ is generated from $\mathcal{U}[0, \epsilon_{t-1}^\star]$, and $L$ uniformly in $\{1, \ldots, L_{\max}\}$. We discuss how to choose $\epsilon_{t-1}^\star$. The very first value $\epsilon_0^\star$ is given in Section 4. Our approach is




(a) (b)

Figure 1: Tempering of a normal distribution to a shifted and correlated normal distribution in dimension 10 (see the example in Section 4.1 for more details). Left: The normalized and weighted squared jumping distance (z-axis) as a function of $\epsilon$ (y-axis) and $L$ (x-axis) for the temperature 0.008. Right: Variation of the difference in energy $\Delta E$ as a function of $\epsilon$ for the same temperature. The values of $L$ are randomized. Based on an SMC sampler with an HMC kernel based on $N = 1{,}024$ particles.

motivated by the upper bound in (2). If for different step sizes $\hat{\epsilon}_t^i$ and different momenta and positions $(p_t^i, \tilde{x}_{t-1}^i)$ for $i \in \{1{:}N\}$ we observe $|\Delta E_t^i| = |H(\hat{p}_t^i, \hat{x}_t^i) - H(p_t^i, \tilde{x}_{t-1}^i)|$, this information may be used to fit a model of the form $|\Delta E_t^i| = f(\hat{\epsilon}_t^i) + \xi_t^i$, where $\xi_t^i$ is assumed to be such that $\forall i, \mathbb{E}[\xi_t^i] = 0, \mathrm{Var}[\xi_t^i] = \sigma^2 < \infty$ and $f : \mathbb{R}^+ \to \mathbb{R}^+$. We may then choose $\epsilon^\star$ so that $f(\epsilon^\star) = |\log 0.9|$. This ensures that the acceptance rate of the HMC kernel stays close to 90%, following suggestions in Betancourt et al. (2014).

In particular we suggest the model $f(\hat{\epsilon}_t^i) = \alpha_0 + \alpha_1(\hat{\epsilon}_t^i)^2$, and we minimize the sum of absolute errors $\sum_{i=1}^N |\xi_t^i|$ w.r.t. $(\alpha_0, \alpha_1)$, which amounts to a median regression. Compared to least squares this approach is more robust to fluctuations in energy, which typically occur when $\epsilon$ approaches its stability limit, as illustrated in Figure 1b.

### Construction of a random distribution for $(\epsilon, L)$

Algorithm 6 describes how to generate values $(\epsilon, L)$ during the second stage. These values are sampled from the weighted empirical distribution supported by the values $(\hat{\epsilon}_t^i, \hat{L}_t^i)$ obtained during the first stage, with weights given by the performance metric (6). We visualize this metric as a function of $(\epsilon, L)$ in Figure 1a.

### Range of values for $L$

During the first stage of our pre-tuning procedure, $L$ is generated uniformly within $\{1, \ldots, L_{\max}\}$. The quantity $L_{\max}$ is initialized to some user-chosen value ($L_{\max} = 100$ in our simulations). Whenever a large proportion of the $L_t^i$ generated by Algorithm 6 is close to $L_{\max}$, we increase $L_{\max}$ by a small amount (5 in our simulations). Similarly,

**Algorithm 6:** (PR) Pre-tuning of the HMC kernel.

**Input**: Resampled particles $\tilde{x}_{t-1}^i$, $i \in 1{:}N$, HMC flow $\widehat{\Phi}$ (targeting $\pi_{t-1}$), $\epsilon_{t-1}^\star$
**Result**: Sample of $(\epsilon_t^i, L_t^i)$, $i \in 1{:}N$, upper bound $\epsilon_t^\star$

**foreach** $i \in 1{:}N$ **do**
  Sample $\hat{\epsilon}_t^i \sim \mathcal{U}[0, \epsilon_{t-1}^\star]$ and $\hat{L}_t^i \sim \mathcal{U}\{1{:}L_{\max}\}$;
  Sample $p_t^i \sim \mathcal{N}(0_d, \mathbf{M}_{t-1})$;
  Apply the leapfrog integration: $(\hat{p}_t^i, \hat{x}_t^i) \leftarrow \widehat{\Phi}_{\hat{\epsilon}_t^i, \hat{L}_t^i}(p_t^i, \tilde{x}_{t-1}^i)$;
  Calculate $\Delta E_t^i$ and $\tilde{\Lambda}(\tilde{x}_{t-1}^i, \hat{x}_t^i)$
Calculate $\epsilon_t^\star$ based on the quantile regression of $\Delta E_t^i$ on $\hat{\epsilon}_t^i$ $\forall i \in 1{:}N$;
Sample $(\epsilon_t^i, L_t^i) \sim \text{Categorical}(w_t^i, \{\hat{\epsilon}_t^i, \hat{L}_t^i\})$, where $w_t^i \propto \tilde{\Lambda}(\tilde{x}_{t-1}^i, \hat{x}_t^i)$ $\forall i \in 1{:}N$;

whenever a large proportion of these values are far away from $L_{\max}$, we decrease $L_{\max}$ by some small amount.

## 3.4 Discussion of the tuning procedures

**Comparison of the two algorithms** The difference between the two procedures consists in the pre-tuning phase at each intermediate step of the sampler. On one hand, pre-tuning makes the SMC sampler more costly per intermediate step. On the other hand this approach makes the sampler more robust to sudden changes in the sequence of distributions. We illustrate this point in our numerical experiments. Both of the suggested tuning procedures have computational costs linear in the number of particles $N$, in line with the other operations in the SMC sampler.

**Other approaches to tuning HMC within SMC** One could try to maximize the squared jumping distance as a function of the position of the particle, based on the associated values of $(\epsilon, L)$. However, learning optimal parameters for each position appears challenging, possibly harder than the original Monte Carlo problem at hand. In line with Girolami and Calderhead (2011) one could use a position dependent mass matrix that would take more information about the target into account, for instance with higher-order derivatives.

Returning to the choice of $(\epsilon, L)$ one could use Bayesian optimization (Snoek et al., 2012), based on the performance of $(\epsilon_{t-1}^i, L_{t-1}^i)$ at the previous iteration. This idea would amount to a parallel version of Mohamed et al. (2013). However, it is not clear how this approach would behave if the underlying distributions evolve over time. Avoiding a pre-tuning step reduces the computational load at the expense of making the sampler potentially less robust. Moreover, the approach of Fearnhead and Taylor (2013) already explores the hyperparameter space adaptively, without additional model specifications. If framed as a bandit problem, fixing over time a grid of possible values $(\epsilon, L)$ could be problematic if the grid misses relevant parts of the hyperparameter space.

**Extensions** The procedure based on pre-tuning can be adapted to random walk MH (RWMH) or MALA (Metropolis adjusted Langevin) kernels. In the first case we may use median regression to find an upper bound for the scale such that the acceptance rate is close to 23.4% (Roberts et al., 1997). In the second case one may target an acceptance rate of 57.4% (Roberts and Rosenthal, 1998). Recall also that MALA kernels may be viewed as HMC kernels with $L = 1$. It recently came to our knowledge that the work of Salomone et al. (2018) also uses a pre-tuning approach for MCMC kernels within SMC samplers. A notable difference is that Salomone et al. (2018) focus on finding a single tuning parameter rather than a distribution.

# 4 Experiments

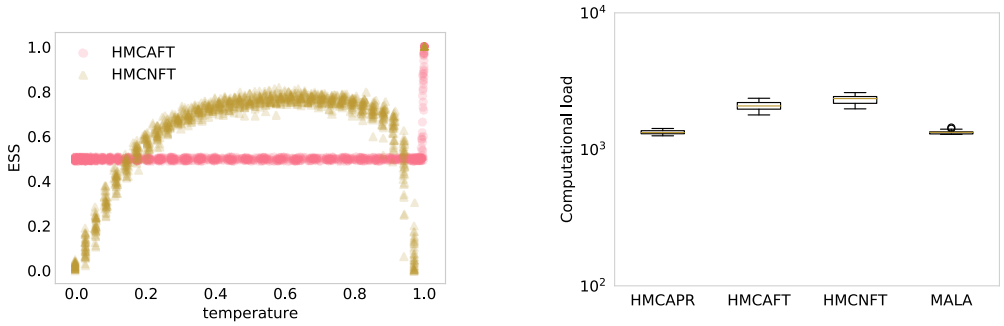
Our experiments highlight the importance of adapting SMC samplers, in particular the parameters of their Markov kernels. Specifically, we try to answer the following questions. How important is it to adapt (a) the number of temperature steps and (b) the number of move steps? (c) Does our tuning procedure of HMC kernels provide reasonable values of $(\epsilon, L)$ compared to other tuning procedures of HMC? (d) To what extent does HMC within an SMC sampler scale with the dimension and may be applied to real data applications? (e) How robust are SMC samplers to multimodality? We compare adaptive (A) and non-adaptive (N) versions of HMC-based SMC samplers, where the adaptation may be using either the FT approach (Fearnhead and Taylor, 2013) or the PR (pre-tuning) approach. We include in our comparison SMC samplers based on random walk (RW) and MALA kernels using FT adaptation. We call our algorithms accordingly: i.e. HMCAFT stands for an SMC sampler using HMC kernels, which are adapted using the FT procedure.

In all the samplers under consideration the mass matrix $\mathbf{M}_t$ is set to the diagonal of the covariance obtained at the previous iteration. Unless otherwise stated, the number of particles is set to $N = 1{,}024$ and resampling is triggered when the ESS drops below $N/2$. The computational load of each sampler is defined as the number of gradient evaluations, plus the number of likelihood evaluations. Note that this is a conservative choice as computations of the likelihood and the gradient often involve some common operations. Most comparisons are in terms of adjusted variance or adjusted mean squared error (MSE), by which we mean variance or MSE multiplied by computational load.

The HMC-based samplers are initialized with uniform draws of $\epsilon$ on $[0, 0.1]$ and of $L$ on $\{1, \ldots, 100\}$. The MALA and RW-based samplers are initialized with uniform draws of the scale in $[0, 1]$. The initial mass matrix is set to the identity. All samplers choose the number of move steps using Algorithm 3. Code for reproducing the figures is available at `https://github.com/alexanderbuchholz/hsmc`.

## 4.1 Tempering from an isotropic Gaussian to a shifted correlated Gaussian

We first consider a tempering sequence that starts at $\pi_0 = \mathcal{N}(0_d, I_d)$, and finishes at $\pi_T = \mathcal{N}(\mu, \Xi)$, where $\mu = 2 \times \mathbf{1}_d$ for different $d$. For the covariance we set the off-diagonal




(a) (b)

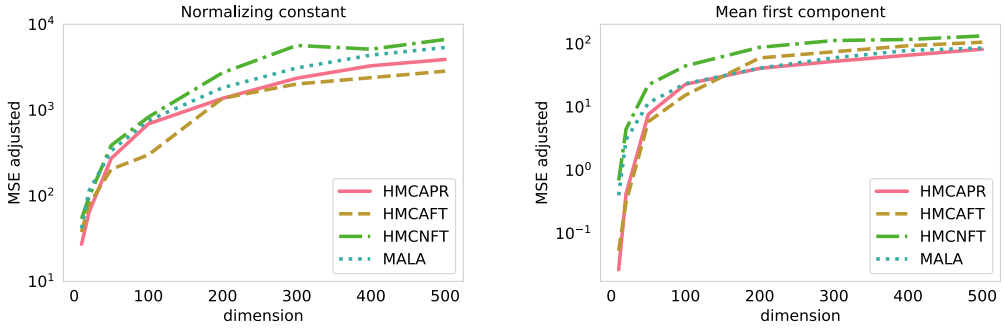
Figure 2: ESS and temperature steps in dimension $d = 500$ (Figure 2a) and computational load (number of gradient and density evaluations, Figure 2b).

correlation to 0.7 and the marginal variances to elements of the equally-spaced sequence $\tilde{\Xi} = [0.1, \cdots, 10]$. We get the covariance $\Xi = \operatorname{diag} \tilde{\Xi}^{1/2} \operatorname{corr}(X) \operatorname{diag} \tilde{\Xi}^{1/2}$. This toy example is challenging due to the different length scales of the variance, the correlation and the shifted mean of the target. The true mean, variance and normalizing constants being available, we can report the mean squared error (MSE) of the estimators. We use normalized importance weights and thus $Z_T/Z_0 = 1$.

We compare the following SMC samplers: MALA, HMCAFT and HMCAPR, according to the denomination laid out in the previous section. We add to the comparison HMCNFT, an SMC sampler using adaptive FT-based HMC steps, but where the sequence of temperatures is fixed a priori to an equi-spaced sequence, the length of which is set according to the number of temperatures chosen adaptively during one run of HMCAFT. Figure 2a plots the ESS as a function of the temperature, for dimension $d = 500$, for algorithms HMCAFT and HMCNFT. Figures 2b and 3 compare the SMC samplers in terms of computational load and adjusted MSE (i.e. MSE times the computational load) for the normalizing constant and the target expectation of the first component. The results for other components are similar to the results for the first component, and not shown here.

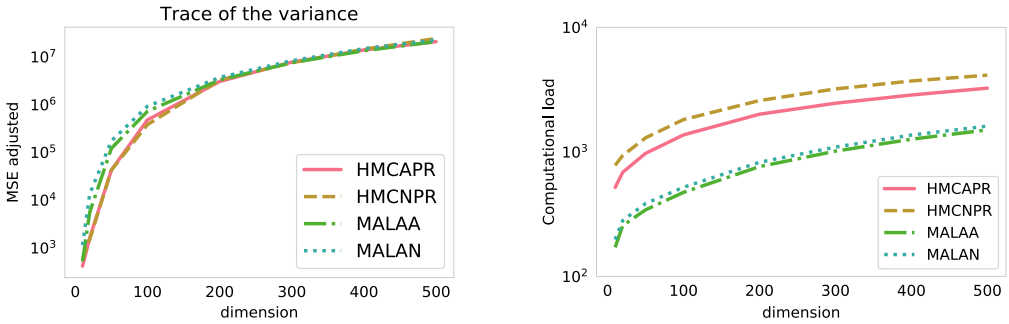
A first observation is that it seems useful to adapt the sequence of temperatures: HMCNFT is outperformed by all the other algorithms. Secondly there is no clear ranking between the other samplers. HMC-based samplers, and particularly HMCAFT, do perform better than the MALA-based sampler for the normalizing constant, but the picture is less clear for the posterior expectation. We remark that MALA kernels may be competitive even in 500 dimensions.

In a second step, we consider the impact of adapting the number of move steps of the samplers. We restrict the comparison to a MALA-based sampler that uses FT tuning and an HMC-based sampler that uses PR tuning. For the non-adaptive samplers (N) the number of move steps is fixed to the average number of move steps obtained from the run of an adaptive sampler. The temperatures are chosen adaptively. We consider




(a) (b)

Figure 3: Mean squared error of the normalizing constant (Figure 3a) and of the first mean (Figure 3b), multiplied by computational cost. Based on 40 repetitions of the samplers with $N = 1{,}024$ particles.




(a) (b)

Figure 4: Mean squared error of the trace of the estimated variance (Figure 4a) multiplied by computational cost (Figure 4b). Based on 40 repetitions of the samplers with $N = 1{,}024$ particles.

the task of estimating the trace of the variance $\Xi$. The results are shown in Figure 4a and the computational cost is shown in Figure 4b. The adaptive samplers seem to offer a similar trade-off in terms of MSE versus computational load.

In order to assess the performance of the two tuning procedures (FT and PR), we compare the tuning parameters obtained at the final stage of our SMC samplers (HMCAFT and HMCAPR) with those obtained from the following MCMC procedures: NUTS (Hoffman and Gelman, 2014) and the adaptive MCMC algorithm of Mohamed et al. (2013). NUTS iteratively tunes the mass matrix $\mathbf{M}$, the number of leapfrog steps $L$ and the step size $\epsilon$ in order to achieve a high ESJD (expected squared jumping distance). The adaptive HMC sampler of Mohamed et al. (2013) uses Bayesian optimization in order to find values of $(\epsilon, L)$ that yield a high ESJD. For this purpose we run HMC-

| Dimension | SMC HMCAPR | SMC HMCAFT | SMC MALA | NUTS | adaptive HMC |
|---|---|---|---|---|---|
| 10 | 50.64 | 61.03 | 9.89 | 59.88 | 134.70 |
| 50 | 174.64 | 255.98 | 30.56 | 204.34 | 190.67 |
| 200 | 639.35 | 989.97 | 85.22 | 1,281.06 | 927.30 |
| 500 | 1,556.27 | 1,311.60 | 154.05 | 2,210.44 | 1,731.04 |

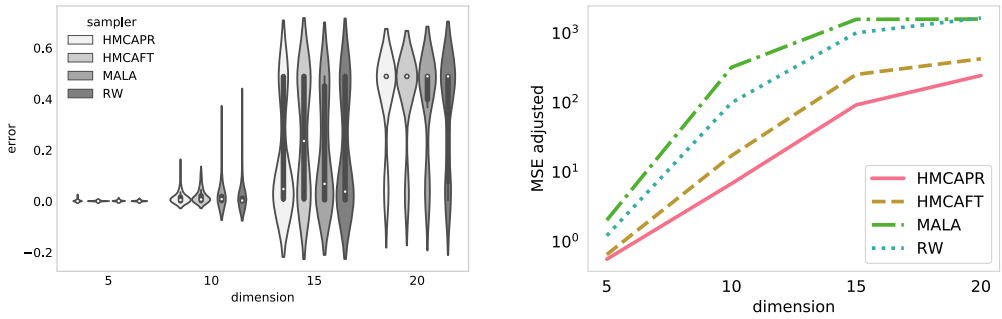
Table 1: Average squared jumping distance for the Gaussian target in the first example, based on 40 runs. Results based on 1,024 samples for the SMC samplers. For the MCMC samplers 2,000 states were generated, with the first 1,000 states discarded as burn-in.

based SMC samplers and record the achieved ESJD of the HMC kernel at the final distribution $\pi_T$. NUTS and the adaptive HMC sampler are run directly on the final target distribution. For NUTS we use the implementation available in Stan; for the adaptive HMC sampler we reimplemented the procedure of Mohamed et al. (2013). After convergence of the tuning procedures, we run the chain for 2,000 iterations and discard 1,000 samples as burn-in. The ESJD is calculated on the remaining 1,000 states. The results are shown in Table 1. We see that the two HMC-based SMC samplers, NUTS and the adaptive HMC sampler achieve an ESJD of the same order of magnitude. Thus, our tuning procedure gives values of $(\epsilon, L)$ that yield an ESJD comparable to other existing procedures.

## 4.2 Tempering from a Gaussian to a mixture of two correlated Gaussians

The aim of the second example is to assess the robustness of SMC samplers with respect to multimodality. We temper from the prior $\pi_0 = \mathcal{N}(\mu_0, 5I_d)$ towards a mixture of shifted and correlated Gaussians $\pi_T = 0.3\mathcal{N}(\mu, \Xi_1) + 0.7\mathcal{N}(-\mu, \Xi_2)$, where $\mu = 4 \times \mathbf{1}_d$ and we set the off-diagonal correlation to 0.7 for $\Xi_1$ and to 0.1 for $\Xi_2$. The variances are set to elements of the equally spaced sequence $\tilde{\Xi}_j = [1, \cdots, 2]$ for $j = 1, 2$. The covariances $\Xi_j$ are based on the same formula as in the first example. To make the example more challenging we set $\mu_0 = \mathbf{1}_d$, making the starting point biased towards one of the two modes. We assess performance by evaluating the signs of the particles via the statistic $T_i := d^{-1} \sum_{j=1}^{d} \mathbb{1}\{\text{sign}\, x_j^i = +1\}$, where $x_j^i$ is the $j$-th component of the $i$-th particle. We would expect 30% of the signs to be positive ($1/N \sum_{i=1}^{N} T_i \approx 0.3$) if the modes were correctly recovered. We define a measure of error as the squared deviation from this value. We consider the following SMC samplers: MALA, RW, HMCAFT, and HMCAPR. All the samplers choose adaptively the number of move steps and the temperatures.

Figure 5b shows that the two HMC-based samplers yield a lower error of the recovered modes adjusted for computation in moderate dimensions. In terms of the recovery of the modes all samplers behave comparably, see Figure 5a. Nevertheless, as the dimension of the problem exceeds 20 all samplers tend to concentrate on one single mode. This problem may be solved in this case by initializing the sampler with a wider distribution, but in general such a fix relies on some knowledge of the location of the modes.




(a) (b)

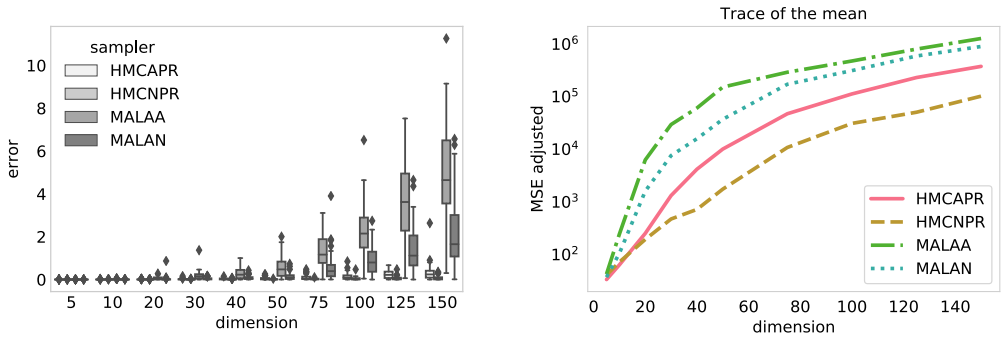
Figure 5: Left: Violin plots of $1/N \sum_{i=1}^{N} T_i - 0.3$. Right: Mean squared error of the proportion of recovered modes adjusted for the computation (based on 40 runs).

Consequently, SMC samplers are robust to multimodality only in small dimensions and care must be taken. That said, HMC-based samplers seem slightly more robust to multimodality; see e.g. results for $d = 10$ in Figure 5a. Interested readers are referred to Mangoubi et al. (2018) for the performance of HMC versus RWMH on multimodal targets.

## 4.3 Tempering from an isotropic Student distribution to a shifted correlated Student distribution

We next study the tempering sequence from a Student distribution $\pi_0 = \mathcal{T}_3(0_d, I_d)$ with $\nu = 3$ degrees of freedom to a shifted and correlated Student with $\nu = 10$, i.e. $\pi_T = \mathcal{T}_\nu(\mu, \Xi)$. The mean and scale matrix are as in Section 4.1. This example presents the challenge that the target $\pi_T$ has heavy tails. We vary the dimension between $d = 5$ and $d = 150$. The temperature steps are chosen such that an ESS of 90% is maintained. The adaptive samplers (A) adjust the number of move steps according to Algorithm 3. For the non-adaptive samplers (N) the number of move steps is fixed to the average number of move steps obtained over a complete run of an adaptive sampler. The MALA-based sampler uses FT tuning, the HMC-based sampler uses our pre-tuning (PR) approach.

The HMC-based samplers perform better when it comes to estimating the normalizing constant (see Figure 6a) and the mean (see Figure 6b). Both samplers using MALA tend to perform poorly for the normalizing constant as dimension increases. Here we observe that the adaptation of the number of move steps can have some negative impact on performance. We suspect that this is due to the heavy tails of the target, as this did not occur in our first example. Furthermore Livingstone et al. (2016) show that an HMC kernel cannot be geometrically ergodic when the target is heavy-tailed. Thus, setting the number of move steps to a fixed large value may be beneficial for heavy-tailed targets.




(a) (b)

Figure 6: Figure 6a shows the squared error of the estimator of the normalizing constant. Figure 6b shows the squared error of the sum of the mean components over different dimensions adjusted for computation. The results are based on 100 runs of the samplers with $N = 1{,}024$ particles.

## 4.4 Binary regression posterior

We next consider a Bayesian binary regression. We observe $J$ vectors $z_j \in \mathbb{R}^d$ and $J$ outcomes $y_j \in \{0, 1\}$. We assume $y_j \sim \text{Bernoulli}(r(z_j^T \beta))$ where $r : \mathbb{R} \mapsto [0, 1]$ is a link function. The parameter of interest is $\beta \in \mathbb{R}^d$, with prior $\beta \sim \mathcal{N}(0_d, I_d)$. With the logit link $r : x \to \exp(-x)/(1 + \exp(-x))$ we obtain a logistic regression with unnormalized log posterior given by

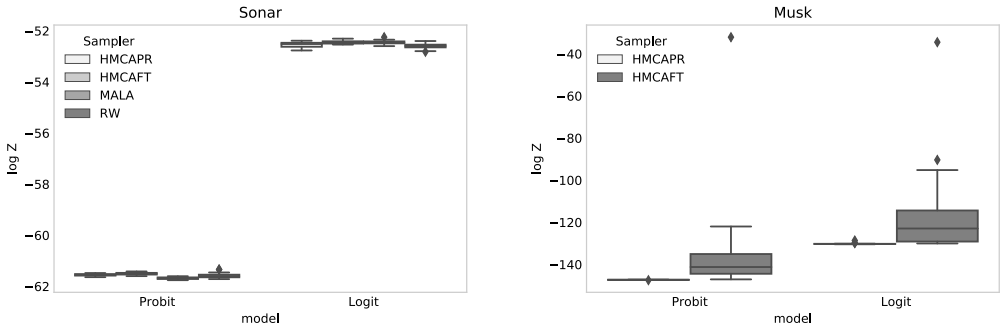
$$\gamma(\beta) = \sum_{j=1}^{J} \left[(y_j - 1) z_j^T \beta - \log(1 + \exp(-z_j^T \beta)\right] - 1/2 \left\|\beta\right\|_2^2 .$$

With a link function given by the cumulative distribution function of a standard Gaussian, denoted by $\Phi$, we obtain a probit regression, with unnormalized log posterior

$$\gamma(\beta) = \sum_{j=1}^{J} \left[y_j \log \Phi(z_j^T \beta) + (1 - y_j) \log\left(1 - \Phi(z_j^T \beta)\right)\right] - 1/2 \left\|\beta\right\|_2^2 .$$

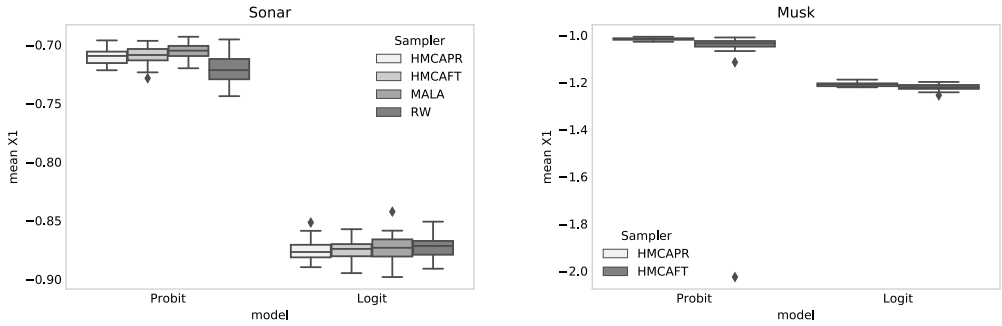
We set $\pi_0$ as a Gaussian approximation of the posterior obtained by Expectation Propagation (Minka, 2001; Chopin and Ridgway, 2017). We consider two datasets available in the UCI (University of California, Irvine) machine learning repository: sonar ($d = 61$ covariates, $J = 208$ observations) and musk ($d = 95$, $J = 476$, after removing a few covariates that are highly correlated with the rest). In both cases an intercept is added and the predictors are standardized to be centered with unit variance.

We compare the following SMC samplers: RW, MALA, HMCAFT, HMCAPR. Figure 7 illustrates the estimation of marginal likelihoods, which may be used for model choice, and Figure 8 shows the estimation of the posterior expectation of the first com-




(a) (b)

Figure 7: Normalizing constants for probit and logit regression models, based on 40 runs. Left: sonar dataset, with 61 predictors. Right: musk dataset, with 95 predictors.




(a) (b)

Figure 8: Estimated first mean for probit and logit regression. Left: sonar dataset, with 61 predictors. Right: musk dataset, with 95 predictors.

ponent. The results for other components are similar and not shown. For all samplers we adapt the number of move steps and the temperatures.

The four samplers perform similarly on the sonar dataset, while they perform differently on the musk dataset. In the latter case, the MALA-based and RW-based samplers did not complete after 48 hours of running, so we had to remove them from the comparison. This was due to the slow mixing of the Markov kernels, resulting in a large number of steps in the while loop in Algorithm 3. In contrast, the two HMC-based samplers took less than 45 minutes to complete. In addition, FT adaptation leads to a high variance of the normalizing constant estimator.

Table 2 reports the logarithm of the adjusted variances for the four considered samplers, the two considered datasets, the two considered models (logit and probit) and a varying number of particles for the sonar dataset. By and large, HMCAPR seems

| | | Normalizing constant | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | Logit | | | | Probit | | | |
| Dataset | Dim | HMCAPR | HMCAFT | MALA | RW | HMCAPR | HMCAFT | MALA | RW |
| Sonar, $N = 2^{10}$ | 61 | 4.183 | **3.489** | 5.776 | 6.459 | 3.315 | **2.98** | 5.915 | 6.081 |
| Sonar, $N = 2^{13}$ | 61 | 2.685 | **2.233** | 4.384 | 5.362 | **2.788** | 3.686 | 4.883 | 5.285 |
| Musk | 95 | **6.09** | 11.519 | – | – | **6.596** | 8.62 | – | – |
| | | Mean first component | | | | | | | |
| | | Logit | | | | Probit | | | |
| Dataset | Dim | HMCAPR | HMCAFT | MALA | RW | HMCAPR | HMCAFT | MALA | RW |
| Sonar, $N = 2^{10}$ | 61 | -3.409 | **-3.644** | -0.978 | -0.875 | **-3.86** | -3.842 | -1.706 | -0.604 |
| Sonar, $N = 2^{13}$ | 61 | -5.482 | **-5.792** | -3.668 | -3.255 | **-5.688** | -3.346 | -3.413 | -2.961 |
| Musk | 95 | **-1.643** | -1.147 | – | – | **-0.633** | 0.014 | – | – |

Table 2: Inefficiency of the estimators of the normalizing constant and the expectation of the first component; based on 40 runs. Smaller is better. The inefficiency is the log of the variance multiplied by the mean number of gradient and likelihood evaluations. For the RWMH sampler the variance is adjusted by the mean number of likelihood evaluations. The best performing sampler for a particular model and number of particles is in bold.

the most robust approach: it often gives either the smallest adjusted variances, or a value close to the smallest one.

### 4.5 Log Gaussian Cox process model

In order to illustrate the performance of HMC-based SMC samplers in high dimensions we consider the log Gaussian Cox process model as in Girolami and Calderhead (2011), applied to the Finnish pine saplings dataset. The dataset consists of the position of 126 trees. The aim is to recover the latent process $X \in \mathbb{R}^{d \times d}$ from the realization of a Poisson process $Y = (y_{j,k})_{j,k}$ with intensity $m\Lambda(j,k) = m\exp(x_{j,k})$, where $m = 1/d^2$ and $X = (x_{j,k})_{j,k}$ is a Gaussian process with mean $\mathbb{E}[X] = \mu \times \mathbf{1}_{d \times d}$ and covariance function $\Sigma_{(j,k)(j',k')} = \sigma^2 \exp(-\delta(j,j',k,k')/(d\beta))$, where $\delta(j,j',k,k') = \sqrt{(j-j')^2 + (k-k')^2}$. The posterior density of the model is given as

$$p(x|y,\mu,\sigma^2,\beta) \propto \left\{ \prod_{j,k}^{d} \exp(y_{j,k}x_{j,k} - m\exp(x_{j,k})) \right\} \times \exp\left\{ -1/2(x-\mu)^T \Sigma^{-1}(x-\mu) \right\},$$

where the second factor is the Gaussian prior density.

Following the results of Christensen et al. (2005) we fix $\beta = 1/33$, $\sigma^2 = 1.91$ and $\mu = \log(126) - \sigma^2/2$. We vary the dimension of the problem from $d = 100$ to $d = 16,384$ by considering different discretizations. We consider three SMC samplers: MALA, HMCAFT and HMCAPR. The starting distribution is the prior. Figure 9b shows that the HMC-based samplers estimate well the normalizing constant, even for a large dimension $d$. Moreover, we estimate the sum of the marginal expectations throughout different dimensions with relatively low variance, see Figure 9a. We omitted the simulation for the MALA-based samplers, as the simulation took excessively long in dimension 4,096 due to slow mixing of the kernel and the while loop in Algorithm 3. The estimated posterior mean of the latent field for dimension 900 is plotted in Figure 9c.

Table 3 reports adjusted variances for the different samplers. We see that the MALA-based sampler is less competitive as the dimension increases. Regarding adaptation, FT outperforms PR, especially in high dimensions.

## 5 Discussion

Our experiments indicate that the relative performance of HMC kernels within SMC depends on the dimension of the problem. For low to medium dimensions, RW and MALA are much faster, and tend to perform reasonably well. On the other hand, for high dimensions, HMC kernels outperform RW and MALA kernels, sometimes significantly.

The key to good performance of SMC samplers, based on HMC or other kernels, is to adaptively tune the Markov kernels used in the propagation step. We have considered two approaches. On posterior distributions with reasonable correlations between components, our adaptation of Fearnhead and Taylor (2013) works best. Our approach based on pre-tuning of the HMC kernels is more robust to changes in the intermediate

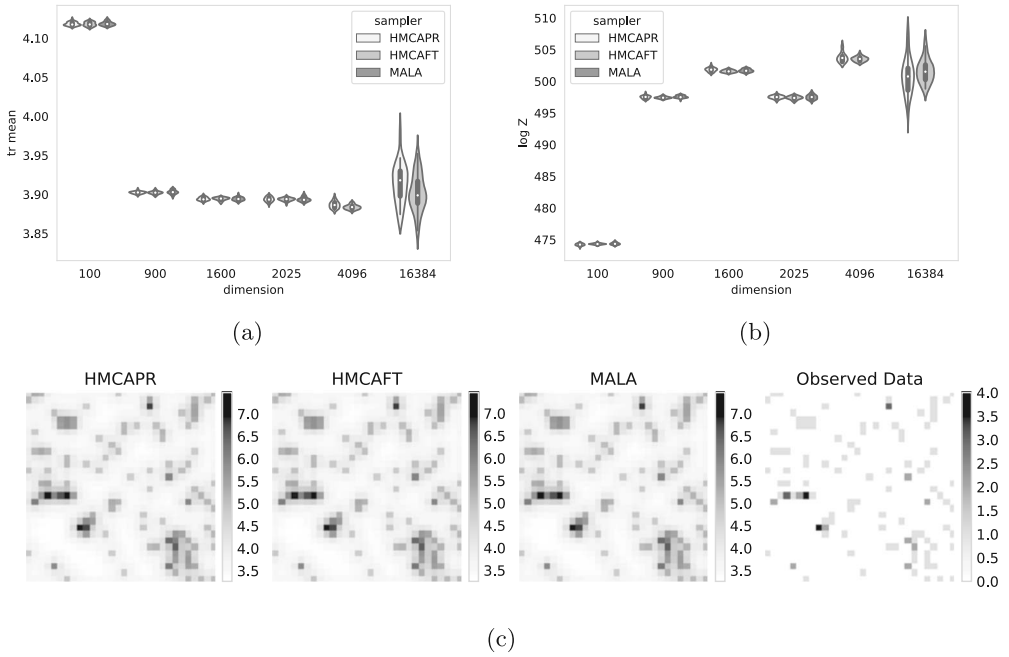



Figure 9: Tempering from a Gaussian prior to the posterior in a log Gaussian Cox process model. Figure 9a illustrates the estimated sum of the posterior marginal expectations. Figure 9b illustrates the estimation of the normalizing constant. Figure 9c illustrates the recovered posterior mean of the process in dimension 900.

| | Normalizing constant | | | Mean first component | | |
|---|---|---|---|---|---|---|
| Dim | HMCAPR | HMCAFT | MALA | HMCAPR | HMCAFT | MALA |
| 100 | 2.292 | 2.03 | 2.933 | -6.4 | -5.613 | -5.559 |
| 400 | 3.296 | 2.255 | 3.812 | -5.895 | -5.913 | -4.765 |
| 900 | 3.89 | 2.776 | 4.373 | -6.192 | -6.141 | -4.276 |
| 1,600 | 4.735 | 3.226 | 5.224 | -5.217 | -6.046 | -4.162 |
| 2,500 | 4.5 | 4.072 | 6.246 | -4.405 | -5.636 | -3.476 |
| 4,096 | 7.055 | 5.071 | – | -3.2 | -4.701 | – |
| 16,384 | 10.002 | 8.864 | – | 0.538 | 0.142 | – |

Table 3: Inefficiency of the normalizing constant estimators and for the first mean, based on 40 runs. The inefficiency is the log of the variance multiplied by the mean number of gradient and likelihood evaluations. Smaller is better.

targets as illustrated in the binary regression example. This holds in particular when using SMC samplers for normalizing constant estimation. Moreover, we observed that SMC samplers with HMC kernels can scale to high dimensional problems when sensibly tuned. From a practical point of view and if the structure of the posterior is unknown the pre-tuning approach may be more prudent.

## References


Agapiou, S., Papaspiliopoulos, O., Sanz-Alonso, D., Stuart, A., et al. (2017). "Importance sampling: intrinsic dimension and computational cost." *Statistical Science*, 32(3): 405–431. MR3696003. doi: https://doi.org/10.1214/17-STS611. 748

Andrieu, C. and Thoms, J. (2008). "A tutorial on adaptive MCMC." *Statistics and computing*, 18(4): 343–373. MR2461882. doi: https://doi.org/10.1007/s11222-008-9110-y. 752

Beskos, A., Jasra, A., Kantas, N., and Thiery, A. (2016). "On the convergence of adaptive sequential Monte Carlo methods." *The Annals of Applied Probability*, 26(2): 1111–1146. MR3476634. doi: https://doi.org/10.1214/15-AAP1113. 745, 748

Beskos, A., Pillai, N., Roberts, G., Sanz-Serna, J.-M., and Stuart, A. (2013). "Optimal tuning of the hybrid Monte Carlo algorithm." *Bernoulli*, 19(5A): 1501–1534. MR3129023. doi: https://doi.org/10.3150/12-BEJ414. 745, 746, 751

Betancourt, M. (2016). "Identifying the optimal integration time in Hamiltonian Monte Carlo." *arXiv preprint* arXiv:1601.00225. 746, 752

Betancourt, M., Byrne, S., and Girolami, M. (2014). "Optimizing the integrator step size for Hamiltonian Monte Carlo." *arXiv preprint* arXiv:1411.6669. MR3644666. 746, 751, 755

Bou-Rabee, N. and Sanz-Serna, J. M. (2018). "Geometric integrators and the Hamiltonian Monte Carlo method." *Acta Numerica*, 27: 113–206. MR3826507. doi: https://doi.org/10.1017/s0962492917000101. 751

Burda, M. and Daviet, R. (2018). "Hamiltonian sequential Monte Carlo with application to consumer choice behavior." Working Papers tecipa-618, University of Toronto, Department of Economics. URL https://ideas.repec.org/p/tor/tecipa/tecipa-618.html 745

Carpenter, B., Gelman, A., Hoffman, M., Lee, D., Goodrich, B., Betancourt, M., Brubaker, M., Guo, J., Li, P., and Riddell, A. (2017). "Stan: A Probabilistic Programming Language." *Journal of Statistical Software, Articles*, 76(1): 1–32. 752

Chopin, N. (2002). "A sequential particle filter method for static models." *Biometrika*, 89(3): 539–552. MR1929161. doi: https://doi.org/10.1093/biomet/89.3.539. 745, 746, 748, 752

Chopin, N. and Ridgway, J. (2017). "Leave Pima Indians alone: binary regression as a benchmark for Bayesian computation." *Statistical Science*, 32(1): 64–87. MR3634307. doi: https://doi.org/10.1214/16-STS581. 762

Chopin, N., Rousseau, J., and Liseo, B. (2013). "Computational aspects of Bayesian spectral density estimation." *Journal of Computational and Graphical Statistics*, 22(3): 533–557. MR3173730. doi: https://doi.org/10.1080/10618600.2013.785293. 746

Christensen, O. F., Roberts, G. O., and Rosenthal, J. S. (2005). "Scaling limits for the transient phase of local Metropolis–Hastings algorithms." *Journal of the Royal*

*Statistical Society: Series B (Statistical Methodology)*, 67(2): 253–268. MR2137324. doi: https://doi.org/10.1111/j.1467-9868.2005.00500.x. 765

Creutz, M. (1988). "Global Monte Carlo algorithms for many-fermion systems." *Physical Review D*, 38(4): 1228. 751

Daviet, R. (2018). "Inference with Hamiltonian sequential Monte Carlo simulators." *arXiv preprint* arXiv:1812.07978. 745

Del Moral, P., Doucet, A., and Jasra, A. (2006). "Sequential Monte Carlo samplers." *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(3): 411–436. MR2278333. doi: https://doi.org/10.1111/j.1467-9868.2006.00553.x. 745, 746, 748

Del Moral, P., Doucet, A., and Jasra, A. (2007). "Sequential Monte Carlo for Bayesian Computation." *Bayesian Statistics*, (8): 1–34. 748

Duane, S., Kennedy, A. D., Pendleton, B. J., and Roweth, D. (1987). "Hybrid Monte Carlo." *Physics letters B*, 195(2): 216–222. MR3960671. doi: https://doi.org/10.1016/0370-2693(87)91197-x. 745

Fearnhead, P. and Taylor, B. M. (2013). "An adaptive sequential Monte Carlo sampler." *Bayesian Analysis*, 8(2): 411–438. MR3066947. doi: https://doi.org/10.1214/13-BA814. 745, 746, 748, 752, 753, 754, 756, 757, 765

Friel, N. and Pettitt, A. N. (2008). "Marginal likelihood estimation via power posteriors." *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 70(3): 589–607. MR2420416. doi: https://doi.org/10.1111/j.1467-9868.2007.00650.x. 748

Girolami, M. and Calderhead, B. (2011). "Riemann manifold Langevin and Hamiltonian Monte Carlo methods." *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(2): 123–214. MR2814492. doi: https://doi.org/10.1111/j.1467-9868.2010.00765.x. 752, 756, 765

Gorham, J. and Mackey, L. (2015). "Measuring sample quality with Stein's method." In *Advances in Neural Information Processing Systems*, 226–234. 749

Gunawan, D., Kohn, R., Quiroz, M., Dang, K.-D., and Tran, M.-N. (2018). "Subsampling sequential Monte Carlo for static Bayesian models." *arXiv preprint* arXiv:1805.03317. 745

Hairer, E., Lubich, C., and Wanner, G. (2003). "Geometric numerical integration illustrated by the Störmer–Verlet method." *Acta numerica*, 12: 399–450. MR2249159. doi: https://doi.org/10.1017/S0962492902000144. 750

Hairer, E., Lubich, C., and Wanner, G. (2006). *Geometric numerical integration: structure-preserving algorithms for ordinary differential equations*, volume 31. Springer Science & Business Media. 751

Hoffman, M. D. and Gelman, A. (2014). "The No-U-turn sampler: adaptively setting path lengths in Hamiltonian Monte Carlo." *Journal of Machine Learning Research*, 15(1): 1593–1623. MR3214779. 746, 751, 752, 759

Huggins, J. H. and Roy, D. M. (2018). "Sequential Monte Carlo as approximate sampling: bounds, adaptive resampling via $\infty$-ESS, and an application to Particle Gibbs." *Bernoulli*. MR3892330. doi: https://doi.org/10.3150/17-bej999. 748

Jasra, A., Paulin, D., and Thiery, A. H. (2015). "Error bounds for sequential Monte Carlo samplers for multimodal distributions." *arXiv preprint* arXiv:1509.08775. MR3892321. doi: https://doi.org/10.3150/17-bej988. 745

Jasra, A., Stephens, D. A., Doucet, A., and Tsagaris, T. (2011). "Inference for Lévy-Driven Stochastic Volatility Models via Adaptive Sequential Monte Carlo." *Scandinavian Journal of Statistics*, 38(1): 1–22. MR2760137. doi: https://doi.org/10.1111/j.1467-9469.2010.00723.x. 748

Kong, A., Liu, J. S., and Wong, W. H. (1994). "Sequential imputation and Bayesian missing data problems." *Journal of the American statistical association*, 89: 278–288. MR3738474. 748

Kostov, S. (2016). "Hamiltonian sequential Monte Carlo and normalizing constants." *Doctoral thesis, University of Bristol*. URL https://ethos.bl.uk/OrderDetails.do?uin=uk.bl.ethos.702941 745

Leimkuhler, B. and Matthews, C. (2016). *Molecular Dynamics*. Springer. MR3362507. 751

Liu, H., Fan, J., and Liao, Y. (2016). "An overview of the estimation of large covariance and precision matrices." *The Econometrics Journal*, 19(1): C1–C32. MR3501529. doi: https://doi.org/10.1111/ectj.12061. 753

Livingstone, S., Betancourt, M., Byrne, S., and Girolami, M. (2016). "On the geometric ergodicity of Hamiltonian Monte Carlo." *arXiv preprint* arXiv:1601.08057. MR3648031. doi: https://doi.org/10.3150/16-BEJ810. 761

Mangoubi, O., Pillai, N. S., and Smith, A. (2018). "Does Hamiltonian Monte Carlo mix faster than a random walk on multimodal densities?" *arXiv preprint* arXiv:1808.03230. 761

Mangoubi, O. and Smith, A. (2017). "Rapid mixing of Hamiltonian Monte Carlo on strongly log-concave distributions." *arXiv preprint* arXiv:1708.07114. 745

Minka, T. P. (2001). "Expectation propagation for approximate Bayesian inference." In *Proceedings of the Seventeenth conference on Uncertainty in artificial intelligence*, 362–369. Morgan Kaufmann Publishers Inc. 762

Mohamed, S., de Freitas, N., and Wang, Z. (2013). "Adaptive Hamiltonian and Riemann manifold Monte Carlo samplers." *arXiv preprint* arXiv:1302.6182. 746, 752, 756, 759, 760

Murray, L. M., Lee, A., and Jacob, P. E. (2016). "Parallel resampling in the particle filter." *Journal of Computational and Graphical Statistics*, 25(3): 789–805. MR3533638. doi: https://doi.org/10.1080/10618600.2015.1062015. 745

Neal, R. M. (1993). "Bayesian learning via stochastic dynamics." In *Advances in neural information processing systems*, 475–482. 745

Neal, R. M. (2001). "Annealed importance sampling." *Statistics and computing*, 11(2): 125–139. MR1837132. doi: https://doi.org/10.1023/A:1008923215028. 745

Neal, R. M. (2011). "MCMC using Hamiltonian dynamics." *Handbook of Markov Chain Monte Carlo*, 2(11). MR3185067. 745, 749, 751, 752

Pasarica, C. and Gelman, A. (2010). "Adaptively scaling the Metropolis algorithm using expected squared jumped distance." *Statistica Sinica*, 343–364. MR2640698. 751

Ridgway, J. (2016). "Computation of Gaussian orthant probabilities in high dimension." *Statistics and computing*, 26(4): 899–916. MR3515028. doi: https://doi.org/10.1007/s11222-015-9578-1. 748

Roberts, G. O., Gelman, A., and Gilks, W. R. (1997). "Weak convergence and optimal scaling of random walk Metropolis algorithms." *The annals of applied probability*, 7(1): 110–120. MR1428751. doi: https://doi.org/10.1214/aoap/1034625254. 757

Roberts, G. O. and Rosenthal, J. S. (1998). "Optimal scaling of discrete approximations to Langevin diffusions." *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 60(1): 255–268. MR1625691. doi: https://doi.org/10.1111/1467-9868.00123. 757

Salomone, R., South, L. F., Drovandi, C. C., and Kroese, D. P. (2018). "Unbiased and consistent nested sampling via sequential Monte Carlo." *arXiv preprint* arXiv:1805.03924. 757

Schäfer, C. and Chopin, N. (2013). "Sequential Monte Carlo on large binary sampling spaces." *Statistics and Computing*, 1–22. MR3016936. doi: https://doi.org/10.1007/s11222-011-9299-z. 745, 748

Schweizer, N. (2012a). "Non-asymptotic error bounds for sequential MCMC and stability of Feynman-Kac propagators." *arXiv preprint* arXiv:1204.2382. 748

Schweizer, N. (2012b). "Non-asymptotic error bounds for sequential MCMC methods in multimodal settings." *arXiv preprint* arXiv:1205.6733. 745

Sim, A., Filippi, S., and Stumpf, M. P. H. (2012). "Information geometry and sequential Monte Carlo." *arXiv e-prints* arXiv:1212.0764. 748

Snoek, J., Larochelle, H., and Adams, R. P. (2012). "Practical Bayesian optimization of machine learning algorithms." In *Advances in neural information processing systems*, 2951–2959. 756

South, L. F., Pettitt, A. N., and Drovandi, C. C. (2019). "Sequential Monte Carlo Samplers with Independent Markov Chain Monte Carlo Proposals." *Bayesian Analysis*, 14(3): 773–796. MR3960770. doi: https://doi.org/10.1214/18-BA1129. 752

Vats, D., Flegal, J. M., and Jones, G. L. (2015). "Multivariate output analysis for Markov chain Monte Carlo." *arXiv preprint* arXiv:1512.07713. MR3653667. 749

Whiteley, N., Lee, A., and Heine, K. (2016). "On the role of interaction in sequential

Monte Carlo algorithms." *Bernoulli*, 22(1): 494–529. MR3449791. doi: https://doi.org/10.3150/14-BEJ666. 748

Zhou, Y., Johansen, A. M., and Aston, J. A. (2016). "Toward Automatic Model Comparison: An Adaptive Sequential Monte Carlo Approach." *Journal of Computational and Graphical Statistics*, 25(3): 701–726. MR3533634. doi: https://doi.org/10.1080/10618600.2015.1060885. 745, 747, 748

**Acknowledgments**

The first author gratefully acknowledges a GENES PhD scholarship, a DAAD grant for visiting the third author and funding from the EPSRC grant EP/R018561/1. The second author is partly supported by Labex Ecodec (anr-11-labx-0047). The third author gratefully acknowledges support by the National Science Foundation through grants DMS-1712872 and DMS-1844695.