Adaptive Tuning Of Hamiltonian Monte Carlo Within Sequential Monte Carlo

Alexander Buchholz[1], Nicolas Chopin[2], Pierre E. Jacob[3]
[1] MRC Biostatistics Unit, University of Cambridge, [2] ENSAE-CREST, [3] Department of Statistics, Harvard University

Abstract

Sequential Monte Carlo (SMC) samplers form an attractive 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.

1 Introduction

Sequential Monte Carlo (SMC) samplers (Neal, 2001; Chopin, 2002; Del Moral et al., 2006) approximate a target distribution π by sampling particles from an initial distribution π_0, and moving them through a sequence of distributions π_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, with direct advantages on parallel computing devices (Murray et al., 2016). Finally, SMC samplers are more robust to multimodality, see Schweizer (2012b) and 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, that depend oftentimes on some tuning parameters. Choosing these parameters in a sensible manner is challenging and has been 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 HMC (Hamiltonian Monte Carlo). HMC has originally been developed in Physics (Duane et al., 1987), and introduced to the Statistics community by Neal (1993). It has become a standard MCMC tool for sampling distributions with continuously differentiable density functions on $\mathbb{R}^d$ (Neal, 2011). The main appeal of HMC is its better mixing (compared to, say, Metropolis samplers) in high-dimensional problems (Beskos et al., 2013; Mangoubi and Smith, 2017).

This paper compares methods for the automatic tuning of HMC kernels within SMC. A few previous papers and thesis considered using HMC kernels within SMC (Gunawan et al., 2018; Burda and Daviet, 2018; Daviet, 2018; Kostov, 2016), but without focusing on tuning the kernels. In our experience, properly tuning MCMC kernels within SMC has a big impact on performance, and particularly so for HMC kernels. As a matter of fact, calibration of HMC kernels is 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 big advantage of tuning such kernels within SMC is that we have at our disposal a cloud of particles that inform us 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 potential of SMC samplers for inference and model choice in high dimensions, and the improved performance brought by HMC kernels within SMC (relative to random walk and Langevin kernels). We also investigate the importance of adapting the tempering ladder, and the number of move steps for diversifying the particles.

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. Section 5 discusses our results.

2 Background

The methods proposed in this article could apply to generic target distributions, but we will focus on posterior distributions and thus we will use the associated terminology. We consider the problem of calculating expectations of an integrable 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 set on $\mathbb{R}^d$. 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 two algorithms widely used to approximate posterior distributions: Sequential Monte Carlo (SMC) and Hamiltonian Monte Carlo (HMC). These are building blocks for the adaptive SMC procedures discussed in this paper.

2.1 Sequential Monte Carlo samplers

2.1.1 Introducing a sequence of targets: tempering from the prior to the posterior

Sequential Monte Carlo (SMC) approaches the problem of sampling from π by introducing a sequence of intermediate distributions $\pi_0, \cdots, \pi_t, \cdots, \pi_T$ defined on the common measurable target space $(\mathbb{R}^d, \mathcal{B}(\mathbb{R}^d))$ for all t, and such that π_0 is easy to sample from, and $\pi_T = \pi$.

We focus on tempering to construct intermediate distributions, that is $\pi_t(x) \propto p(x)l(y|x)^{\lambda_t}$, where the sequence of exponents λ_t is such that $0 = \lambda_0 < \cdots < \lambda_t < \cdots < \lambda_T = 1$. These exponents do not need to be pre-specified: they 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). Note also that we assume throughout the article that the prior distribution $p(x)$ is a proper probability distribution, from which samples can be drawn.

2.1.2 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: $Z_t = \int_{\mathbb{R}^d} \gamma_t(x)dx$. One way of constructing SMC samplers is as follows. Suppose that at time $t-1$ an equally weighted particle approximation $\left\{\tilde{x}_{t-1}^i\right\}_{i\in 1:N}$ of π_{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 π_{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 $\left\{x_t^i\right\}_{i\in 1:N}$ 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 π_t

is approximated by the set $\left\{x_t^i, w_t^i\right\}_{i\in 1:N}$. After resampling particles according to their weights, one obtains again an equally weighted set $\left\{\tilde{x}_t^i\right\}_{i\in 1:N}$ and the procedure is repeated for the next target distribution π_{t+1}. The resulting algorithm is given in Algorithm 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 as follows:

$$\frac{\sum_{i=1}^N w_t^i \varphi(x_t^i)}{\sum_{i=1}^N w_t^i} \to \mathbb{E}_{\pi_t}\left[\varphi(x)\right]$$

as $N \to +\infty$. The algorithm also returns estimates of the ratios Z_t/Z_{t-1} (and thus of Z_T/Z_0); see line 13. (In the context of tempering, one may use the path sampling identity to derive an alternate estimate of Z_t/Z_{t-1}, as explained in Zhou et al. (2016). However, in our experiments, the two estimates were very close numerically, so we focus on the former estimate from now on.)

The kernels $\mathcal{K}_t$ may be constructed, for example, as Metropolis–Hastings 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 general construction 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 $\mathcal{K}_t^h$ to make this dependence explicit, as in Algorithm 1.

Algorithm 1: SMC sampler with MCMC moves on tempered posteriors

Input: Number of particles N, Initial distribution π_0, target distribution π_T and intermediate distributions $\pi_{t-1}(x) \propto p(x)l(y|x)^{\lambda_{t-1}}$, rule for constructing Markov kernels $\mathcal{K}_t^h$ that are π_{t-1} invariant.

Result: Set of weighted samples $\left\{x_t^i, w_t^i\right\}_{i\in 1:N}$ for $t \in 1:T$ and normalizing constant estimates $\widehat{Z_t/Z_{t-1}}$ for $t \in 1:T$.

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 available 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 for better mixing, see Algorithm 3 ;
10 Choose the next exponent $\lambda_t \in (\lambda_{t-1}, 1]$ based on available 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 $\widehat{\frac{Z_t}{Z_{t-1}}} = N^{-1}\sum_{i=1}^N w_t^i$;
14 Resample particles $\left\{x_t^i, w_t^i\right\}_{i\in 1:N}$ to obtain $\left\{\tilde{x}_t^i\right\}_{i\in 1:N}$;
15 Set $t = t + 1$;

2.1.3 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 λ_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 compared to the usual 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 linked to the χ^2-divergence between the distributions π_{t-1} and π_t or equivalently to the variance of the weights. More precisely 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 χ^2 divergence from π_{t-1} to π_t.

We may choose λ_t by solving (in λ) 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 e.g. Beskos et al. (2016), see also Huggins and Roy (2018); Whiteley et al. (2016). Another approach for choosing the sequence of temperature steps is exposed in Friel and Pettitt (2008).

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

Input: Target value α, likelihood $l(y|x_t^i)$ for the N particles x_t^i, current temperature λ_{t-1}.
Result: Next temperature λ_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 λ_t.

(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 a distribution π, mixing is improved by repeated application of the kernel, for a cost linear 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 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.

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 : ∏_{l=1}^{k} ρ̂_l(j) > α'}/d ≥ 10% do
    Set k ← k + 1;
    Move particle x_{t,k}^i ~ K_t^h(x_{t,k-1}^i, dx) for all i;
    Calculate the correlation ρ̂_k(j) for all j;
```

Instead of using component-wise autocorrelations, one could also draw on recent work on the performance evaluation of MCMC algorithms in multidimensional spaces (Vats et al., 2015) or approaches based on the Stein discrepancy (Gorham and Mackey, 2015).

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 first describe HMC, by following the exposition in Neal (2011), before turning to existing approaches to the tuning of its algorithmic parameters.

2.2.1 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

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

The physical analogy of this quantity is the Hamiltonian, where the first term denotes the potential energy at position x, and the second term denotes the kinetic energy of the momentum p with the mass matrix $\mathbf{M}$. The movement in time of a particle with position x and momentum p can be described via its 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 τ. The solution of this differential equation induces a flow Φ_τ 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 Φ_τ 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 flow is not available and one has to use numerical integration methods instead. One widely used integrator is the Störmer-Verlet or leapfrog integrator (Hairer et al., 2003). The leapfrog integrator is volume preserving and reversible but not energy preserving. It iterates the following updates:

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

where ϵ denotes the step size of the leapfrog integrator. Thus, in order to let the system evolve from τ to $\tau + \kappa$ with $\kappa = L \times \epsilon$ we need to make L steps as described 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 μ on the joint space, with a Metropolis–Hastings step that corrects for the variation in the energy after sampling a random momentum and constructing a numerical flow. Algorithm 4 describes the Markov kernel of HMC.

Algorithm 4: Hamiltonian Monte Carlo algorithm

Input: Gradient function $\nabla_x \mathcal{L}(\cdot)$, initial state x_s, energy function ΔE
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 $x_{s+1} \leftarrow x_s$

2.2.2 Existing approaches to tuning HMC

The error analysis of geometric integration gives insights that allow to find step sizes ϵ that yield stable trajectories. For the leapfrog integrator the error of the energy is

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

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 (ϵ, L). Using the error control in (2), Neal (2011) following Creutz (1988) provides an informal reasoning motivating the scaling of ε as $d^{-1/4}$, at least for targets that factorize into products of d independent components. To maintain a fixed integration time εL 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 ϵ 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 ϵ 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 ϵ 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 in order to obtain reasonable values of ϵ.

A different approach is to choose the parameters of the kernel such that the expected squared jumping distance (ESJD) of the kernel is maximized, see Pasarica and Gelman (2010). The ESJD in one dimension is defined as

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

assuming stationarity. In this sense maximizing the ESJD of a Markov chain is equivalent to minimizing the first order autocorrelation ρ_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 case of HMC this has been discussed by Mohamed et al. (2013) and Hoffman and Gelman (2014). Mohamed et al. (2013) tune the HMC sampler with Bayesian optimization and vanishing adaptation in the spirit of adaptive MCMC algorithms, see Andrieu and Thoms (2008). The ESJD is then maximized as a function of (ϵ, L). Hoffman and Gelman (2014) discuss the ESJD as a criterion for the choice of L. As a general idea the simulation of the trajectories should be stopped when the ESJD starts to decrease. However, this idea has the inconvenience of impacting the reversibility of the chain. This problem is solved by adjusting the acceptance step in the algorithm. The resulting algorithm is called NUTS (Hoffman and Gelman, 2014) and is used in the probabilistic programming language STAN, see Carpenter et al. (2017).

Neal (2011) suggests to use preliminary runs in order to find reasonable values of (ϵ, L) and to randomize the values around the chosen values. The randomization avoids pathological behavior that might occur when (ϵ, L) are selected badly. Other approaches on identifying the optimal trajectory length are discussed in Betancourt (2016).

Another important tuning parameter is the mass matrix $\mathbf{M}$, that is used for sampling 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. This yields the same transformation, see Neal (2011). Recently, Girolami and Calderhead (2011) suggested to use a position dependent mass matrix that takes the local curvature of the target into account. However, the numerical integrator for the Hamiltonian equation needs to be modified in consequence.

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 can be linked to the tuning of MCMC kernels in general. One advantage

of tuning the kernels in the framework of SMC is that information on the intermediate distributions is available in form of the particle approximations. Moreover, different kernel parameters can be assigned to different particles and hence a large number of parameters can be tested in parallel. This idea has been exploited by Fearnhead and Taylor (2013). We build upon their methodology and adjust their approach to the tuning of HMC kernels.

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 pre-tuning 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 the mass matrix $\mathbf{M}$, used for sampling the momentum. Calibrating this matrix based on the particles of the previous iteration allows to exploit the information generated by the particles. In the case of an independent MH proposal this has been used by Chopin (2002) and more recently by South et al. (2019). For the HMC kernel we use the following matrix 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 the particle estimate of the covariance matrix of target π_t. The restriction to a diagonal matrix makes this approach applicable in high dimensions; alternatively a non-diagonal covariance or precision matrix could be estimated in high dimension by assuming sparsity (see e.g. Liu et al., 2016). Thus, the global structure of the target distribution π_{t-1} is taken into account and moves in directions of higher variance are proposed.

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

Suppose we are at line 6 during iteration t of Algorithm 1. We are now interested in choosing the parameters h of the propagation kernel $\mathcal{K}_t^h$.

Fearnhead and Taylor (2013) consider the following ESJD (expected squared jumping distance) criterion:

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

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 covariance matrix of the particles at time $t-1$, but, again, this requires inverting a full matrix, 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 ΔE_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 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 ϵ is perturbed by a small (truncated) Gaussian noise, and L has an equal chance of increasing, decreasing or staying the same. The variance of the Gaussian noise is set to the value used by Fearnhead and Taylor (2013) (in our simulations we found that the tuning was robust to different choices of this value). The resulting algorithm is given in Algorithm 5.

Algorithm 5: (FT) Tuning of the HMC algorithm 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 time 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 as an alternative to the previous approach the following two-stage procedure:

1. We apply an HMC step (with respect to π_{t-1}) to the N current particles; for each particle the value of (ϵ, L) is chosen randomly from a certain uniform distribution (described in next section). We then construct a new random distribution for (ϵ, L) based on the performance of these N steps (Section 3.3.2). The HMC trajectories are then discarded.

2. We apply again an HMC step to the N current particles, except this time (L, ϵ) is generated from the distribution constructed in the previous step.

We now explain this approach in more detail.

3.3.1 Range of values for ϵ

In the first stage of our pre-tuning parameter, ϵ is generated from $\mathcal{U}[0, \epsilon^\star_{t-1}]$, and L from $\mathcal{U}[0, L_{\max}]$. We discuss in this section how to choose $\epsilon^\star_{t-1}$. The value of the very first $\epsilon^\star_0$ is given in Section 4.

Our approach is motivated by the upper bound in (2). If for different step sizes $\hat{\epsilon}^i_t$ and different momenta and positions $(p^i_t, \tilde{x}^i_{t-1})$ for $i \in \{1:N\}$ we observe $\left|\Delta E^i_t\right| = \left|H(\hat{p}^i_t, \hat{x}^i_t) - H(p^i_t, \tilde{x}^i_{t-1})\right|$, this information may be used to fit a model of the form

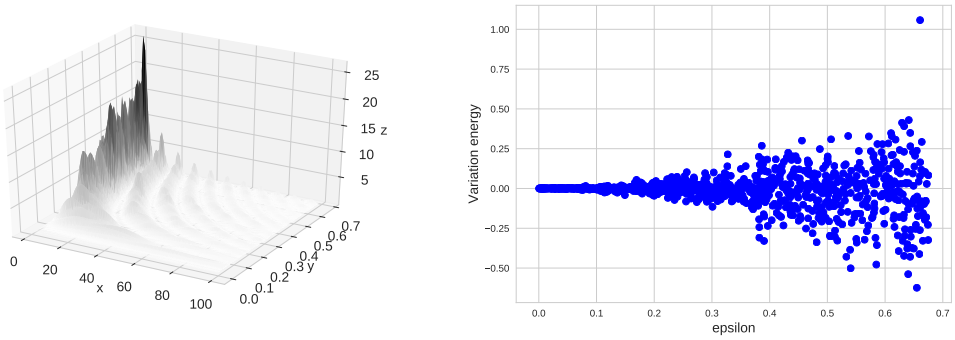
$$\left|\Delta E^i_t\right| = f(\hat{\epsilon}^i_t) + \xi^i_t,$$

where ξ^i_t is assumed to be such that $\forall i, \mathbb{E}\left[\xi^i_t\right] = 0, \mathrm{Var}\left[\xi^i_t\right] = \sigma^2 < \infty$ 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 the suggestions of Betancourt et al. (2014).

In particular we suggest to use the model

$$f(\hat{\epsilon}^i_t) = \alpha_0 + \alpha_1(\hat{\epsilon}^i_t)^2,$$

and we minimize the sum of the absolute value errors $\sum_{i=1}^N \left|\xi^i_t\right|$ w.r.t. (α_0, α_1), which amounts to a median regression. Compared to least squares regression, this approach is more robust to high fluctuations in the energy which typically occur when ϵ approaches its stability limit. We illustrate this point in Figure 1b.



(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 ϵ (y-axis) and L (x-axis) for the temperature 0.008. Right: Variation of the difference in energy ΔE as a function of ϵ 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.

3.3.2 Construction of a random distribution for (ϵ, L)

Algorithm 6 describes how we generate values for (ϵ, L) during the second stage of our pre-tuning procedure. In words, these values are sampled from the weighted empirical distribution that correspond to the N values $(\hat{\epsilon}^i_t, \hat{L}^i_t)$ generated (uniformly) during the first stage, with weights given by

performance metric (6). We visualize this metric as a function of ϵ, L in Figure 1a.

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

Input: Resampled particles $\tilde{x}_{t-1}^i$, $i \in 1:N$, HMC flow $\widehat{\Phi}_{\cdot,\cdot}$ (targeting π_{t-1}), $\epsilon_{t-1}^\star$
Result: Sample of (ϵ_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 ΔE_t^i and $\tilde{\Lambda}(\tilde{x}_{t-1}^i, \hat{x}_t^i)$
Calculate $\epsilon_t^\star$ based on the quantile regression of ΔE_t^i on $\hat{\epsilon}_t^i$ $\forall i \in 1:N$;
Sample $(\epsilon_t^i, L_t^i) \sim \mathcal{C}at\left(w_t^i, \{\hat{\epsilon}_t^i, \hat{L}_t^i\}\right)$, where $w_t^i \propto \tilde{\Lambda}(\tilde{x}_{t-1}^i, \hat{x}_t^i)$ $\forall i \in 1:N$;

3.3.3 Range of values for L

During the first stage of our pre-tuning procedure, L is generated uniformly within range $[0, L_{\max}]$. The quantity $L_{\max}$ is initialized to some user-chosen value (we took $L_{\max} = 100$ 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, whenever a large proportion of these values are far away from $L_{\max}$, we decrease $L_{\max}$ by the same 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 a sudden change 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 performed in the SMC sampler.

Other potential approaches to tuning HMC within SMC One could try to maximize the squared jumping distance as a function of the position of every particle, based on the associated values of (ϵ, L). However, learning optimal parameters for every position might be challenging, possibly harder than the original Monte Carlo problem of interest. In line with Girolami and Calderhead (2011) one could use a position dependent mass matrix that would take the geometry of the target space into account, for instance related to higher-order derivatives of the target probability density function.

Returning to the choice of (ϵ, L) one could use an approach based on Bayesian optimization (Snoek et al., 2012), based on the performance of the $(\epsilon_{t-1}^i, L_{t-1}^i)$ 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. Not using 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 requiring additional model specifications. If framed as a Bandit problem, fixing over time a grid of possible values (ϵ, L) could be problematic if the grid misses relevant parts of the

hyperparameter space. This holds true in particular when using continuous Bayesian optimization, where one typically has to define some box constraints on the underlying space.

Extensions The tuning procedure based on pre-tuning might also be used for tuning random walk (RW) Metropolis 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). (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 with our approach is that Salomone et al. (2018) concentrates 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 (ϵ, 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?

For this purpose we compare adaptive (A) and non-adaptive (N) versions of HMC-based SMC samplers, where the adaptation may be carried out using either the FT approach (our variant of the approach of Fearnhead and Taylor, 2013) or the PR (pre-tuning) approach. We also include in our comparison SMC samplers based on random walk (RW) and MALA kernels, and the FT adaptation procedure. 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 considered SMC samplers, the mass matrix $\mathbf{M}_t$ of the MCMC kernels is set to the diagonal of the covariance matrix obtained at the previous iteration. Unless otherwise stated, the number of particles is $N = 1,024$ and the resampling is triggered when the ESS drops below $N/2$. The computational load of a given 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 the same computations. Most of our comparisons are in terms of adjusted variance, or adjusted MSE (mean squared error), by which we mean: variance (or MSE) times computational load.

All HMC-based samplers are initialized with uniform random draws of ϵ on $[0, 0.1]$ and L on $\{1, 100\}$. The MALA and RW-based samplers are initialized with random draws of the scale parameter on $[0, 1]$. The initial mass matrix is set to the identity for all different samplers. All samplers choose adaptively the number of move steps based on Algorithm 3. Code for reproducing the results shown below is available under `https://github.com/alexanderbuchholz/hsmc`.

4.1 Tempering from an isotropic Gaussian to a shifted correlated Gaussian

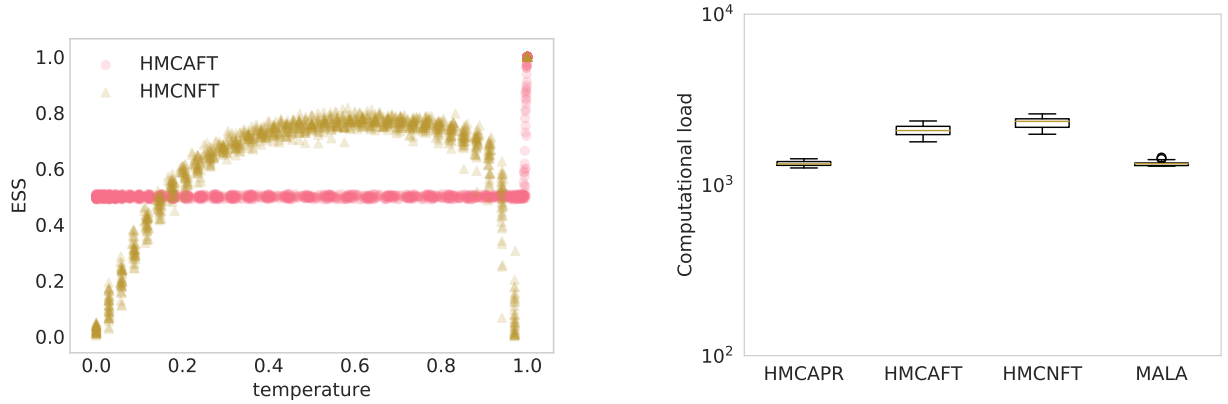
As a first toy example we consider a tempering sequence that starts at an isotropic Gaussian $\pi_0 = \mathcal{N}(0_d, I_d)$, and finishes at a shifted and correlated Gaussian $\pi_T = \mathcal{N}(\mu, \Xi)$, where $\mu = 2 \times \mathbf{1}_d$, for different values of d. For the covariance we set the off-diagonal 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 rather challenging due to the

different length scales of the variance, the correlation and the shifted mean of the target. In this example the true mean, variance and normalizing constants are available. Therefore we report the mean squared error (MSE) of the estimators. We use normalized importance weights and thus $Z_T/Z_0 = 1$.

We first 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 a long equi-spaced sequence (the size of which is set according to the number of temperatures chosen adaptively during one run of HMCAFT).

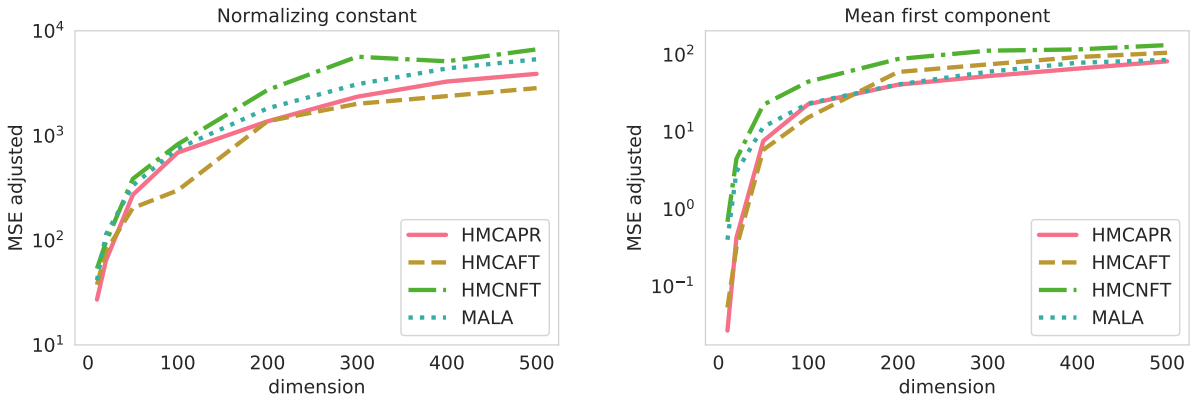
Figure 2a plots the ESS as a function of the temperature, for dimension $d = 500$, for algorithms HMCAFT and HMCNFT. Figures 2b and 3 compare the SMC samplers in terms of computational load (Figure 2b) and adjusted MSE (i.e. MSE times the computational load) for the normalizing constant and the expectation of the first component (with respect to the target). The results for other components (not shown here) are similar to the results for the first component.

A first observation is that it seems useful to adapt the sequence of temperatures: HMCNFT is outperformed by all the other algorithms for both estimates. A second observation is that there is no clear ranking between the three 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 of the first component. It is remarkable that, even in dimensions as high as 500, MALA kernels may be competitive.



(a) (b)

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

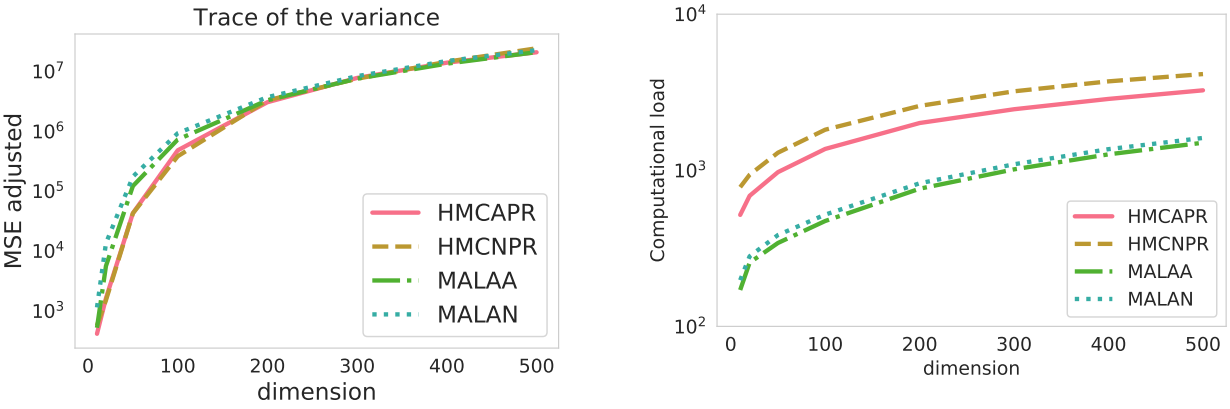


(a) (b)

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

In a second step, we compare 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 a constant number, equal to the average number of move steps over the complete run of an adaptive sampler. The temperatures are chosen adaptively.

We compare the performance of the samplers in estimating the trace of the variance Ξ. The results are shown in Figure 4a and the associated computational cost is shown in Figure 4b. The adaptive samplers seem to offer a similar trade-off in terms of MSE versus computational load.



(a) (b)

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

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 ϵ 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 (ϵ, L) that yield a high ESJD.

For this purpose we run our HMC-based SMC samplers and record the achieved ESJD of the HMC kernel at the final distribution π_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 the convergence of the tuning procedures we run the chain for $2,000$ iterations and discard a burnin of $1,000$ samples. The ESJD is calculated on the remaining $1,000$ iterations of the chain. The results of this comparison are shown in Table 1. We see that both the HMC-based SMC samplers, NUTS and the adaptive HMC tuning achieve an ESJD of the same order of magnitude. Thus, our tuning procedure gives values of (ϵ, L) that yield an ESJD comparable to other existing procedures.

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 of different algorithms for the Gaussian target in the first example (based on 40 runs). The results are based on 1,024 samples for the SMC samplers. For the MCMC samplers (NUTS, adaptive HMC) we used a length of 2,000 states where the first 1,000 states are discarded as burn-in.

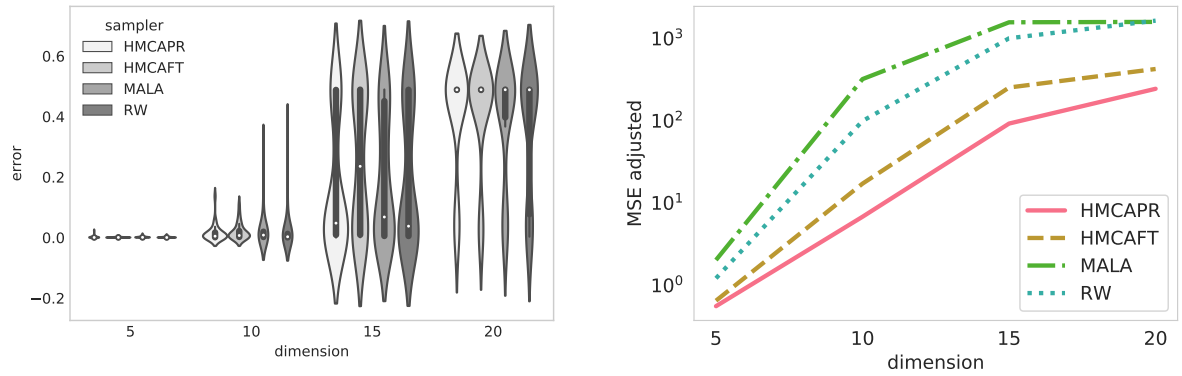
4.2 Tempering from a Gaussian to a mixture of two correlated Gaussians

The aim of our 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 Ξ_1 and to 0.1 for Ξ_2. The variances are set to elements of the equally spaced sequence $\tilde{\Xi}_j = [1, \cdots, 2]$ for $j = 1, 2$. The covariances Ξ_j are based on the same formula as in the first example. In order to make the example more challenging we set $\mu_0 = \mathbf{1}_d$. Thus, the starting point of the sampler is slightly biased towards one of the two modes. We evaluate the performance of the samplers by evaluating the signs of the particles and use therefore the statistic $T_i := 1/d \sum_{j=1}^{d} \mathbb{1}_{\{\text{sign } X_{j,i}=+1\}}$, based on a single particle. We expect a proportion of 30% of the signs to be positive, i.e. $1/N \sum_{i=1}^{N} T_i \approx 0.3$, if the modes are correctly recovered. Our measure of error is based on 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.

As shown by Figure 5b 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 by initializing the sampler with a wider distribution. However, this approach relies on the knowledge of the location of the modes.

Consequently, SMC samplers are robust to multimodality only in small dimensions and care must be taken when using SMC samplers in this setting. That said, HMC-based samplers seem

slightly more robust to multimodality; see e.g. results for $d = 10$ in Figure 5a. See also the recent work of Mangoubi et al. (2018) for the performance of HMC versus RWMH on multimodal targets.

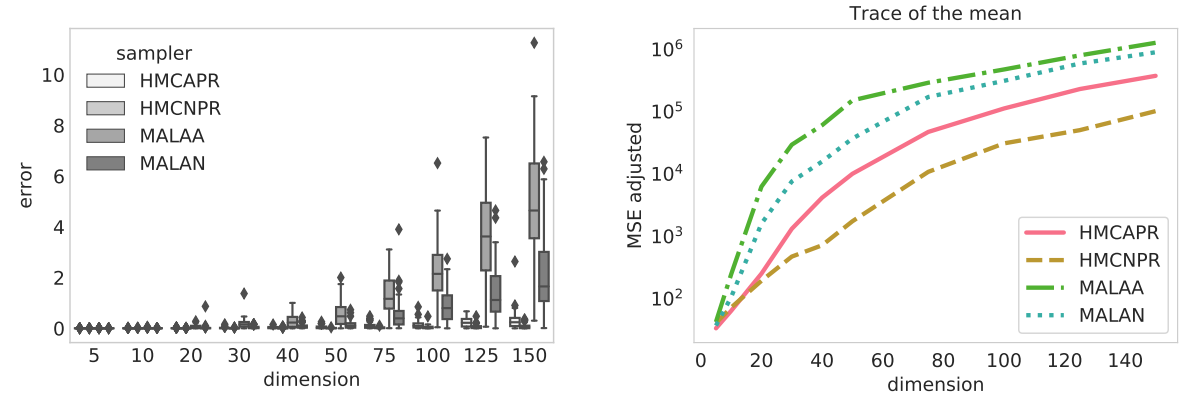


(a) (b)

Figure 5: Left: Violinplot of the error, based on the squared difference 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 of the samplers.

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

As a different challenging toy example we study the tempering from an isotropic Student distribution $\pi_0 = \mathcal{T}_3(0_d, I_d)$ with 3 degrees of freedom towards a shifted and correlated Student distribution with $\nu = 10$ degrees of freedom, i.e. $\pi_T = \mathcal{T}_\nu(\mu, \Xi)$. The mean and scale matrix are set as in the unimodal Gaussian example in 4.1.



(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 trace of the mean over different dimensions adjusted for computation. The results are based on 100 runs of the samplers with $N = 1,024$ particles.

This example is more challenging as the target distribution π_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 attained. 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 a constant number, equal to the average number of move steps over the complete run of an adaptive sampler. The temperatures are chosen adaptively. 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 mean (see Figure 6b) and the normalizing constant (see Figure 6a). Both samplers based on a MALA kernel tend to work poorly in terms of the estimation of the normalizing constant when the dimension increases, see Figure 6a.

In this particular example, we observe that the adaptation of the number of move steps has some negative impact on the performance of the samplers. We suspect that this is due to the heavy tails of the Student distribution: first, this phenomenon did not occur in our first numerical example, where the target had light tails (Gaussian); second, 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. Our approach provides a guideline for finding this number.

4.4 Binary regression posterior

We now consider a Bayesian binary regression model. We observe J vectors $z_j \in \mathbb{R}^d$ and J outcomes $y_j \in \{0, 1\}$. We assume $y_j \sim \mathcal{B}er(r(z_j^T \beta))$ where $r : \mathbb{R} \mapsto [0, 1]$ is a link function. The parameter of interest is $\beta \in \mathbb{R}^d$, endowed with a Gaussian prior, i.e. $\beta \sim \mathcal{N}(0_d, I_d)$.

When using the logit link function $r : x \to \exp(-x)/(1+\exp(-x))$ we obtain a Bayesian logistic regression where the unnormalized log posterior is given as

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

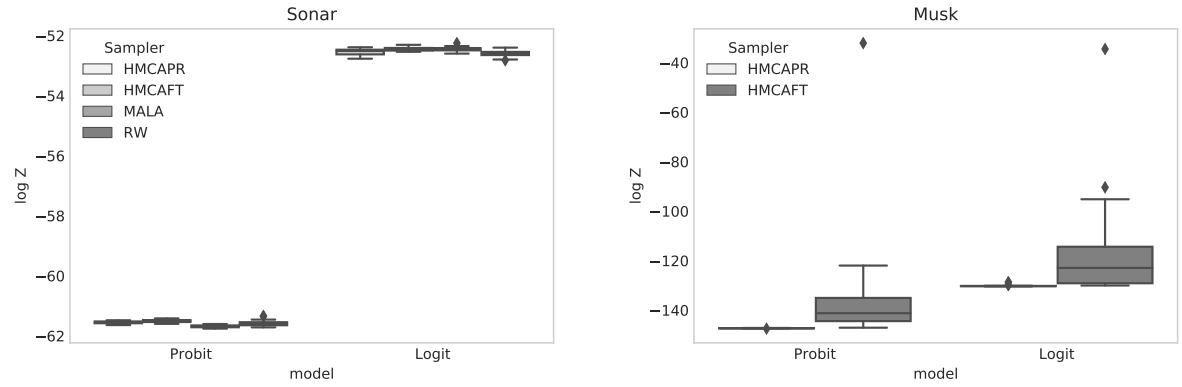
When using as link function the cumulative distribution function of a standard normal distribution, denoted Φ, one obtains the Bayesian Probit regression. In this case the unnormalized log posterior is given as

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

We set π_0 to a Gaussian approximation of the posterior obtained by Expectation Propagation (Minka, 2001) as in Chopin and Ridgway (2017).

We consider two datasets (both available in the UCI 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 normalised (to have mean 0 and variance 1).

We compare the following SMC samplers: RW, MALA, HMCAFT, HMCAPR; see Figure 7 for the estimation of the marginal likelihoods (which may be used to perform model choice) and Figure 8 for the estimation of the posterior expectation of the first component. The results for other components (not shown here) are similar to the results for the first component. For all samplers we adapt the number of move steps and the temperature steps.

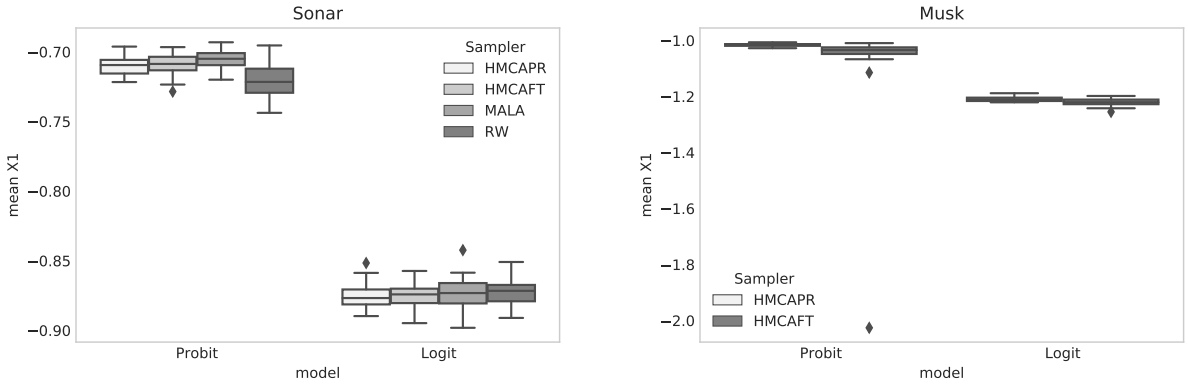


(a) (b)

Figure 7: Normalization constants obtained for the probit and logit regression based on 40 runs of the samplers. Figure 7a corresponds to the normalization constants obtained for the sonar dataset. The posterior has 61 dimensions. Figure 7b corresponds to the musk dataset. The posterior has 95 dimensions.

The four samplers perform similarly on on the sonar dataset, while they perform quite 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. (In contrast, the two HMC-based samplers took less than 45 minutes to complete). In addition, FT adaptation leads to a high variance for the normalising constant.

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



(a) (b)

Figure 8: Estimated mean of the first component of the posterior obtained for the probit and logit regression. Figure 8a corresponds to the sonar dataset. Figure 8b corresponds to the musk dataset.

		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 estimation of the normalizing constant and the mean of the first component based on 40 runs of the different samplers. Smaller is better. The inefficiency is measured as the log of the adjusted variances of the samplers (variance multiplied by the mean number of gradient and likelihood evaluations). For the RWMH sampler we adjust by the number of likelihood evaluations only. The best performing sampler for a particular model and number of particles is highlighted in bold.

4.5 Log Gaussian Cox model

In order to illustrate the advantage of HMC-based SMC samplers in high dimensions we consider the log Gaussian Cox point process model in Girolami and Calderhead (2011), applied to the Finnish pine saplings dataset. This dataset consists of the geographic 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 cumulative mean 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. The estimated posterior mean of the latent field for dimension 900 is plotted in Figure 9c.

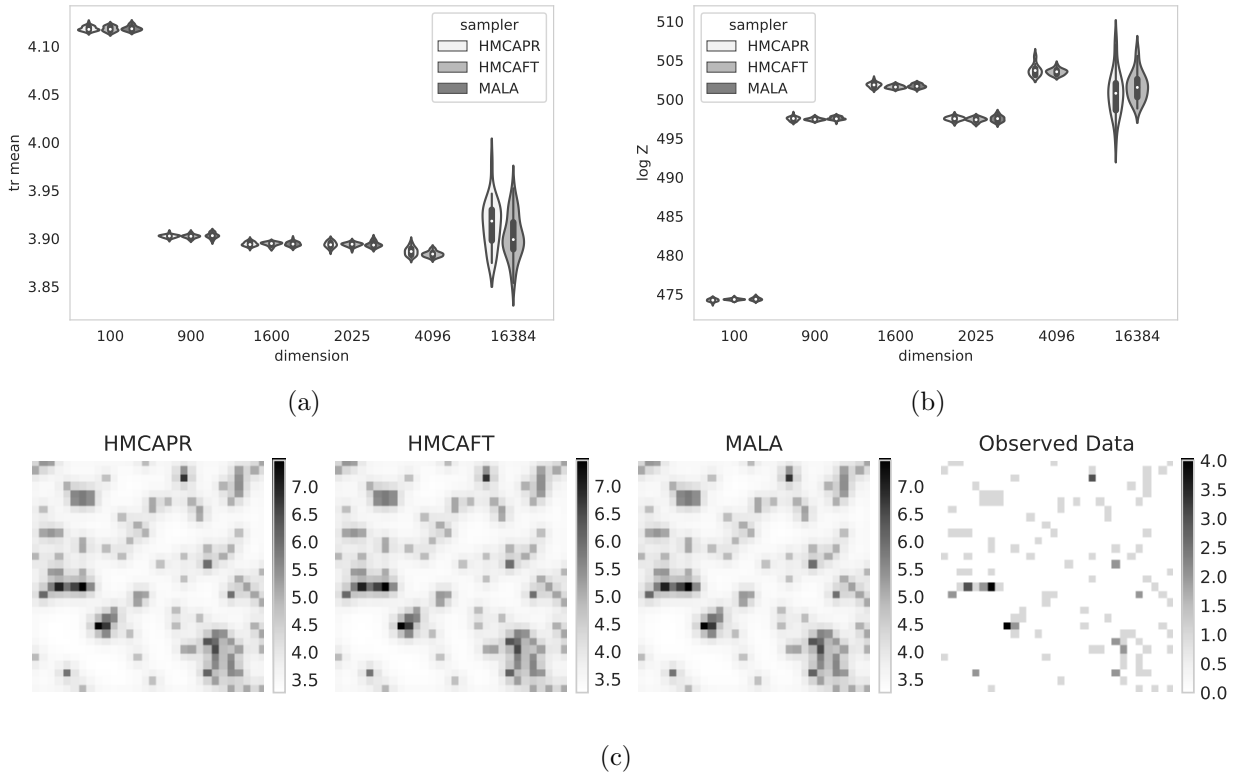


Figure 9: Tempering from a normal prior to the posterior of log Gaussian Cox process over various dimensions. Figure 9a illustrates the estimations of the normalizing constants. Figure 9b illustrates the estimated cumulative posterior mean. Figure 9c illustrates the recovered componentwise posterior mean of the process in dimension 900.

Table 3 reports adjusted variances (variances times computational load) for the considered SMC samplers. We see that the MALA-based sampler is not competitive when the dimension increases. Regarding the adaptation scheme, FT outperforms PR, especially in high dimension.

	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 estimation of the normalizing constant and the mean of the first component based on 40 runs of the different samplers. The inefficiency is measured as the log of adjusted variances of the samplers (variance multiplied by the mean number of gradient and likelihood evaluations). Smaller is better.

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, sometimes significantly, RW and MALA kernels.

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 in this paper. On posterior distributions with reasonable correlation our adaption of the approach by Fearnhead and Taylor (2013) works best. Our approach based on pre-tuning of the HMC kernels is more robust to changes in the subsequent target distributions, as illustrated by our binary regression example. This holds in particular when using SMC samplers for model choice. Moreover, we showed that, when sensibly tuned, SMC samplers with HMC kernels can scale to high dimensional problems. From a practical point of view and if the structure of the posterior is unknown the second approach may be the more prudent choice.

All in all, our methodology provides a principled approach for an automatic adaption of SMC samplers, applicable over a range of various models in different dimensions.

Acknowledgments

The first author gratefully acknowledges a GENES research 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.

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.

Andrieu, C. and Thoms, J. (2008). A tutorial on adaptive MCMC. *Statistics and computing*, 18(4):343–373.

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.

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.

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

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

Bou-Rabee, N. and Sanz-Serna, J. M. (2018). Geometric integrators and the Hamiltonian Monte Carlo method. *Acta Numerica*, 27:113206.

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.

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.

Chopin, N. (2002). A sequential particle filter method for static models. *Biometrika*, 89(3):539–552.

Chopin, N. and Ridgway, J. (2017). Leave Pima Indians alone: binary regression as a benchmark for Bayesian computation. *Statistical Science*, 32(1):64–87.

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.

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.

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

Daviet, R. (2018). Inference with Hamiltonian Sequential Monte Carlo Simulators. *arXiv preprint arXiv:1812.07978*.

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.

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

Duane, S., Kennedy, A. D., Pendleton, B. J., and Roweth, D. (1987). Hybrid Monte Carlo. *Physics letters B*, 195(2):216–222.

Fearnhead, P. and Taylor, B. M. (2013). An adaptive sequential Monte Carlo sampler. *Bayesian Analysis*, 8(2):411–438.

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.

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.

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

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*.

Hairer, E., Lubich, C., and Wanner, G. (2003). Geometric numerical integration illustrated by the Störmer–Verlet method. *Acta numerica*, 12:399–450.

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

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.

Huggins, J. H. and Roy, D. M. (2018). Sequential monte carlo as approximate sampling: bounds, adaptive resampling via ∞-ess, and an application to particle gibbs. *Bernoulli*.

Jasra, A., Paulin, D., and Thiery, A. H. (2015). Error Bounds for Sequential Monte Carlo Samplers for Multimodal Distributions. *arXiv preprint arXiv:1509.08775*.

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.

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.

Kostov, S. (2016). Hamiltonian sequential Monte Carlo and normalizing constants. *Doctoral thesis, University of Bristol*.

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

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.

Livingstone, S., Betancourt, M., Byrne, S., and Girolami, M. (2016). On the geometric ergodicity of Hamiltonian Monte Carlo. *arXiv preprint arXiv:1601.08057*.

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*.

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

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

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

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.

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

Neal, R. M. (2001). Annealed importance sampling. *Statistics and computing*, 11(2):125–139.

Neal, R. M. (2011). MCMC using Hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*, 2(11).

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

Ridgway, J. (2016). Computation of Gaussian orthant probabilities in high dimension. *Statistics and computing*, 26(4):899–916.

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.

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.

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*.

Schäfer, C. and Chopin, N. (2013). Sequential Monte Carlo on large binary sampling spaces. *Statistics and Computing*, pages 1–22.

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

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

Sim, A., Filippi, S., and Stumpf, M. P. H. (2012). Information Geometry and Sequential Monte Carlo. *arXiv e-prints*, page arXiv:1212.0764.

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

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.

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

Whiteley, N., Lee, A., and Heine, K. (2016). On the role of interaction in sequential Monte Carlo algorithms. *Bernoulli*, 22(1):494–529.

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.