The Apogee to Apogee Path Sampler

Chris Sherlock, Szymon Urbas & Matthew Ludkin

To cite this article: Chris Sherlock, Szymon Urbas & Matthew Ludkin (2023): The Apogee to Apogee Path Sampler, Journal of Computational and Graphical Statistics, DOI: 10.1080/10618600.2023.2190784

To link to this article: https://doi.org/10.1080/10618600.2023.2190784

The Apogee to Apogee Path Sampler

Chris Sherlock[1,†,*], Szymon Urbas[1], Matthew Ludkin[2]

[1]Department of Mathematics and Statistics, Lancaster University, UK
[2]Darktrace, Cambridge, UK.

[†]Corresponding author:Chris Sherlock, c.sherlock@lancaster.ac.uk

*All three authors gratefully acknowledge funding through EPSRC grant EP/P033075/1.

Abstract

Amongst Markov chain Monte Carlo algorithms, Hamiltonian Monte Carlo (HMC) is often the algorithm of choice for complex, high-dimensional target distributions; however, its efficiency is notoriously sensitive to the choice of the integration-time tuning parameter. When integrating both forward and backward in time using the same leapfrog integration step as HMC, the set of apogees, local maxima in the potential along a path, is the same whatever point (position and momentum) along the path is chosen to initialise the integration. We present the Apogee to Apogee Path Sampler (AAPS), which utilises this invariance to create a simple yet generic methodology for constructing a path, proposing a point from it and accepting or rejecting that proposal so as to target the intended distribution. We demonstrate empirically that AAPS has a similar efficiency to HMC but is much more robust to the setting of its equivalent tuning parameter, the number of apogees that the path crosses.

Keywords: Leapfrog step, Hamiltonian Monte Carlo, Markov chain Monte Carlo, robustness to tuning.

1 Introduction

Markov chain Monte Carlo (MCMC) is often the method of choice for estimating expectations with respect to complex, high-dimensional targets (e.g. Gilks et al., 1996; Brooks et al., 2011). Amongst MCMC algorithms, Hamiltonian Monte Carlo (HMC, also known as Hybrid Monte Carlo; Duane et al., 1987) is known to offer a performance that scales better with the dimension of the state space than many of its rivals (Neal, 2011b; Beskos et al., 2013).

Given a target density $\pi(x), x \in \mathbb{R}^d$, with respect to Lebesgue measure, and a current position, at each iteration HMC samples a momentum and numerically integrates Hamiltonian dynamics on a potential surface

$$U(x) = -\log \pi(x) \qquad (1)$$

to create a proposal that will either be accepted or rejected. As such, HMC has two main tuning parameters: the numerical integration step size, ϵ, and the total integration time, *T*. Given *T*, guidelines for tuning ϵ have been available for some time (Beskos et al., 2013); however, the integration time itself, is notoriously difficult to tune (e.g. Neal, 2011b), with algorithm efficiency often dropping sharply following only slight changes from the optimum *T* value, and usually exhibiting approximately cyclic behaviour as *T* increases.

The sensitivity is illustrated in the left panel of Figure 1, in which HMC is applied to a modified Rosenbrock distribution (see Section 4.1) of dimension *d* = 40. In the top half of this plot, the efficiency (see (8) for a precise definition), is given as a function of ϵ and the number of numerical integration steps, *L*. The bottom half of the panel shows the analogous plot for a modification of the HMC algorithm (Mackenze, 1989; Neal, 2011b), which we refer to as *blurred HMC* where at each iteration, the actual step-size is sampled (independently of all previous choices) uniformly from the interval $[0.8\epsilon, 1.2\epsilon]$. This step was designed to mitigate the near reducibility of HMC that can occur when *T* is some rational multiple of the integration time required to return close to the starting point, but as is visible from the plots, it also makes the performance of the algorithm more robust to the choice of *T*, and, we have found, often leads to a slightly more efficient algorithm. In both cases, the optimal tuning choice appears as a narrow ridge of roughly constant $T = L\epsilon$. Blurred HMC can be viewed as sampling the integration time *T* uniformly from $[\frac{2}{3}T_*, T_*]$, where $T_* = 1.2L\epsilon$. The approach of using a random integration time to introduce robustness has been extended recently to sampling uniformly from $[0, T_*]$ (Hoffman et al., 2021) and as an exponential variable with an expectation of T_* (Bou-Rabee and Sanz-Serna, 2017).

Motivated by the difficulty of tuning *T*, Hoffman and Gelman (2014) introduces the no U-turn sampler (NUTS). This uses the same numerical integration scheme as standard HMC, the leapfrog step, to integrate Hamiltonian dynamics both forward and backward from the current point, recursively doubling the size of the path until the distance between the points the farthest forward and backward in time stops increasing. Considerable care must be taken to ensure that the intended posterior is targeted, making the algorithm relatively complex; however, the no U-turn sampler is the default engine behind the popular STAN package (Stan Development Team, 2020), which has a relatively straightforward user interface.

Integration of Hamiltonian dynamics can be thought of as describing the position and momentum of a particle as it traverses the potential surface *U*. During its journey, provided *T* is not too small, the particle will reach one or more local maximum, or *apogee*, in the potential. The leapfrog scheme creates a path which is a discrete set of points rather than a continuum, and so (with probability 1) apogees occur between consecutive points in the path; however, they are straightforward to detect. We call the set of points (positions and momenta) between two apogees a *segment*. The discrete dynamics using the leapfrog step share several properties with the true dynamics, including the following: if we take the position and momentum of any point along the path, and integrate forward and backward for appropriate lengths of time we will create exactly the same path, and hence the same set of apogees and the same set of segments. This *invariance* is vital to the correctness and flexibility of the algorithm presented in this article.

In Section 3 we present the *Apogee to Apogee Path Sampler* (AAPS). Like HMC, AAPS is straightforward to implement and has two tuning parameters, one of which is an integration step size, ϵ. As with the no U-turn sampler, given a current point (position and momentum), AAPS uses the leapfrog step to integrate forwards and backwards in time. However, the integration stops when the path contains the segment in which the current point lies as well as K additional segments, where K is a user-defined tuning parameter. A point is then proposed from this set of $K + 1$ segments and either accepted or rejected. The positioning of the current segment within the $K + 1$ and the accept-reject probability are chosen precisely so that the algorithm targets the intended density. The invariance of the path to the starting position and momentum leads to considerable flexibility in the method for proposing a point from the set of segments, which in turn allows us to create an algorithm which enjoys a similar efficiency to HMC yet is extremely robust to the choice of ϵ and K. These properties are evident from the right-hand panel of Figure 1 and analogous plots for other distributions in Section 4.1. The robustness arises mainly from the proposal scheme; see the end of Section 3.3. The definition of K, however, also naturally caters to intrinsic properties of the target, such as its eccentricity, with any externally imposed length (or time) scale largely irrelevant; the theoretical analysis of Section 3.6 makes this explicit in a simplified setting.

Section 2 describes Hamiltonian dynamics and HMC. The AAPS is detailed in Section 3, and is compared empirically against HMC and the no U-turn sampler in Section 4. We conclude in Section 5 with a discussion.

2 Hamiltonian dynamics and Hamiltonian Monte Carlo

2.1 Hamiltonian dynamics

The position, x, and momentum, p, of a particle on a frictionless potential surface $U(x)$ evolve according to Hamilton's equations:

$$\frac{\mathrm{d}x}{\mathrm{d}t} = M^{-1}p \quad \text{and} \quad \frac{\mathrm{d}p}{\mathrm{d}t} = -\nabla_x U. \qquad (2)$$

For a real object, *M* is the mass of the object, a scalar, but the properties of the equations themselves that we will require hold more generally, when *M* is a symmetric, positive-definite mass matrix. The choice of *M*, whether for HMC or AAPS, is discussed at the end of Section 3.5. We define $z_t = (x_t, p_t)$ and the map ϕ_t which integrates the dynamics forwards for a time *t*, so $z_t = \phi_t(z_0)$. The map, ϕ_t has the following fundamental properties (e.g. Neal, 2011b):

1. It is deterministic.
2. It has a Jacobian of 1.
3. It is skew reversible: $\phi_t(x_t, -p_t) = (x_0, -p_0)$.
4. It preserves the total energy $H(x,p) = U(x) + \frac{1}{2} p^\top M^{-1} p$.

Except in a few special cases, the dynamics are intractable and must be integrated numerically, with a user-chosen time step which we will denote by ϵ. The default method for Hamiltonian Monte Carlo, is the method which will be used throughout this article, the *leapfrog* step; the leapfrog step itself is detailed in Appendix A. Throughout the main text of this article we denote the action of a single leapfrog step of length ϵ on a current state *z* as $\text{LeapFrog}(z;\epsilon)$. The leapfrog step satisfies Properties 1-3 above (see Appendix A), but it does not preserve the total energy.

Consider using *L* leapfrog steps of size $\epsilon = t/L$ to approximately integrate the dynamics forward for time *t*. Since each individual leapfrog step satisfies Properties 1-3, so does the composition of *L* such steps, which we denote $\hat{\phi}_t(z_0;\epsilon)$.

2.2 Hamiltonian Monte Carlo

Hamiltonian Monte Carlo (HMC) creates a Markov chain which has a stationary distribution of $\pi(x) = \exp\{-U(x)\}$. Given a current position, x^{curr}, and tuning parameters ϵ and *L*, a single iteration of the algorithm proceeds as follows:

1. Sample a momentum $p_0 \sim \text{N}(0, M)$ and set $z_0 = (x^{\text{curr}}, p_0)$.
2. For *i* in 1 to *L*:

    - $z_i = \text{LeapFrog}(z_{i-1};\epsilon)$.

3. Let $z^{\text{prop}} = z_L$ and set $\alpha = 1 \wedge \tilde{\pi}(z^{\text{prop}}) / \tilde{\pi}(z^{\text{curr}})$.
4. With probability *α*, $z^{\text{curr}} \leftarrow z^{\text{prop}}$; else $z^{\text{curr}} \leftarrow z^{\text{curr}}$.

Here, with $\rho(p)$ denoting the density of the $\mathrm{N}(0, M)$ random variable,

$$\tilde{\pi}(z) \equiv \tilde{\pi}(x, p) = \pi(x)\rho(p) = \exp\{-H(x, p)\}. \qquad (3)$$

If x^{curr} is in its stationary distribution then $\tilde{\pi}$ is the density of $z_0 = (x^{\text{curr}}, p_0)$.

3 The Apogee to Apogee Path Sampler

3.1 Apogees and segments

The left panel of Figure 2 shows *L* = 50 leapfrog steps of size $\epsilon = 0.1$ from a current position, *x*$_0$ simulated randomly from a two-dimensional posterior $\pi(x)$ (with contours of *U*(*x*) shown), and momentum *p*$_0$ simulated from a $\mathrm{N}(0, I_2)$ distribution. Different symbols and colours are used along the path, with both of these changing from step *l* to step *l* + 1 if and only if

$$p_l^{\top} M^{-1} \nabla U(x_l) > 0 \text{ and } p_{l+1}^{\top} M^{-1} \nabla U(x_{l+1}) < 0. \qquad (4)$$

Intuitively, condition (4) indicates when the “particle” has switched from moving “ uphill” to moving “downhill” with respect to the potential surface *U*(*x*). By (2), $p^{\top} M^{-1} \nabla U \equiv \mathrm{d}x / \mathrm{d}t \cdot \nabla U \equiv \mathrm{d}U / \mathrm{d}t$, so (4) indicates a local maximum in $U(x_t)$ between *x*$_l$ and x_{l+1}.

Between such a pair of points at *l* and *l* + 1 there is a hypothetical point, *a* with $l < a < l + 1$ where the particle’s potential has reached a local maximum, which we call an *apogee*. Under the exact, continuous dynamics, this point would, of course, be realised, but under the discretised dynamics the probability of this is 0. We call each of the realised sections of the path between a pair of consecutive apogees (*i.e.*, each portion with a different colour and symbol in Figure 2) a *segment*. Each segment consists of the time-ordered list of position and momentum at each point between two consecutive apogees.

Instead of integrating forward for *L* steps, one can imagine integrating both forwards and backwards in time from $z_0 = (x_0, p_0)$ until a certain number of apogees have been found. The right pane of Figure 2 shows the segment to which the current point belongs, which we denote $\mathcal{S}_0(z_0)$, together with two segments forward and one segment backward. We denote the *j*th segment forward by $\mathcal{S}_j(z_0)$ and the *j*th segment backward by $\mathcal{S}_{-j}(z_0)$. We abbreviate the ordered collection of segments from $\mathcal{S}_a(z_0)$ to $\mathcal{S}_b(z_0)$ as $\mathcal{S}_{a:b}(z_0)$. Thus, the right panel of Figure 2 shows the positions from $\mathcal{S}_{-1:2}(z_0)$. For a particular point $z' = (x', p') \in \mathcal{S}_{a:b}$, we denote the segment to which it belongs by $\mathcal{S}_{\#}(z'; z_0)$.

The following *segment invariance property* is vital to both the simplicity and correctness of our algorithm. For any $K \geq 0$ and $c \in \{0, 1, \ldots, K\}$ set $a = -c$ and $b = K - c$. For any *z* and any $z' \in \mathcal{S}_{a:b}(z)$,

$$\mathcal{S}_{a':b'}(z') \equiv \mathcal{S}_{a:b}(z), \text{ where } a' = -c', b' = K - c' \text{ and } c' = c + \mathcal{S}_{\#}(z'; z). \qquad (5)$$

The quantities a', b' and c' correspond to *a*, *b* and *c* but from the point of view of z' rather than *z*. For the right panel of Figure 2, for example, picking any $z' = (x', p')$ from $\mathcal{S}_1(z_0), \mathcal{S}_{-2:1}(z')$ would give the same ordered set of segments as illustrated in the figure. This is because the numerical integration scheme is deterministic and skew reversible, so the apogees would all occur in exactly the same positions with exactly the same (up to a possible sign flip) momenta.

3.2 The AAPS algorithm

We now introduce our algorithm, the Apogee to Apogee Path Sampler, AAPS. The algorithm requires a weight function $w : \mathbb{R}^{4d} \to [0, \infty)$, where *d* is the dimension of the target. Weight functions are investigated in more detail in Sections 3.3, but for now it might be helpful keep in mind the simplest that we consider: $w(z, z') = \tilde{\pi}(z')$.

Given a step-size ϵ, a non-negative integer, *K*, a mass matrix, *M* and a current position x^{curr}, one iteration of AAPS proceeds as follows:

1. Sample a momentum $p_0 \sim \mathrm{N}(0, M)$ and set $z^{\text{curr}} = z_0 = (x^{\text{curr}}, p_0)$.
2. Simulate *c* uniformly from $\{0, 1, \ldots, K\}$; set $a = -c$ and $b = K - c$.
3. Create $\mathcal{S}_{a:b}$ by leapfrog stepping <u>forward</u> from (x_0, p_0) and then <u>backward</u> from (x_0, p_0).
4. Propose z^{prop} w.p. $\propto w(z^{\text{curr}}, z^{\text{prop}}) 1(z^{\text{prop}} \in \mathcal{S}_{a:b}(z^{\text{curr}}))$.
5. With a probability of

$$\alpha = 1 \wedge \frac{\tilde{\pi}(z^{\text{prop}}) w(z^{\text{prop}}, z^{\text{curr}}) \sum_{z \in \mathcal{S}_{a:b}(z^{\text{curr}})} w(z^{\text{curr}}, z)}{\tilde{\pi}(z^{\text{curr}}) w(z^{\text{curr}}, z^{\text{prop}}) \sum_{z \in \mathcal{S}_{a:b}(z^{\text{curr}})} w(z^{\text{prop}}, z)} \qquad (6)$$

   set $z^{\text{curr}} \leftarrow z^{\text{prop}}$ else $z^{\text{curr}} \leftarrow z^{\text{curr}}$.

6. Discard p^{curr} and retain x^{curr}.

The Metropolis-Hastings formula (6) arises because, out of the allowable proposals once *c* has been chosen, the probability of proposing z^{prop} is $q(z^{\text{prop}} \mid z^{\text{curr}}) = w(z^{\text{curr}}, z^{\text{prop}}) / \sum_{z \in \mathcal{S}_{a:b}} w(z^{\text{curr}}, z)$.

Proposition 1. *The* AAPS *algorithm satisfies detailed balance with respect to the extended posterior* $\tilde{\pi}(x, p)$.

Proof. Step 1 preserves $\tilde{\pi}$ because *p* is independent of *x* and is sampled from its marginal.

It will be helpful to define the system from the point of view of starting at z^{prop}. Let a', b' and c' be as defined in (5), but with $z' = z^{\text{prop}}$. Then, since $\mathcal{S}_{\#}(z^{\text{prop}}; z^{\text{curr}}) = -\mathcal{S}_{\#}(z^{\text{curr}}; z^{\text{prop}})$,

$$-c \leq 0 \leq K - c, -c \leq \mathcal{S}_{\#}(z^{\text{prop}}; z^{\text{curr}}) \leq K - c \Leftrightarrow -c' \leq \mathcal{S}_{\#}(z^{\text{curr}}; z^{\text{prop}}) \leq K - c', -c' \leq 0 \leq K - c'.$$

Equivalently, $1(c \in \{0, \ldots, K\}) 1(z^{\text{prop}} \in \mathcal{S}_{a:b}(z^{\text{curr}})) \equiv 1(c' \in \{0, \ldots, K\}) 1(z^{\text{curr}} \in \mathcal{S}_{a':b'}(z^{\text{prop}}))$.
Moreover, segment invariance (5) is equivalent to $z \in \mathcal{S}_{a:b}(z^{\text{curr}}) \Leftrightarrow z \in \mathcal{S}_{a':b'}(z^{\text{prop}})$.

The resulting chain satisfies detailed balance with respect to $\tilde{\pi}$ because

$$\tilde{\pi}(z^{\text{curr}}) \times \mathbb{P}(c) \times \mathbb{P}\left(\text{propose } z^{\text{prop}} \mid c\right) \times \mathbb{P}\left(\text{accept } z^{\text{prop}} \mid \text{proposed}, c\right)$$

is

$$\tilde{\pi}(z^{\text{curr}}) \times \frac{1(c \in \{0,\ldots,K\})}{K+1} \times \frac{w(z^{\text{curr}}, z^{\text{prop}}) 1(z^{\text{prop}} \in \mathcal{S}_{a:b}(z^{\text{curr}}))}{\sum_{z \in \mathcal{S}_{a:b}(z^{\text{curr}})} w(z^{\text{curr}}, z)}$$
$$\times 1 \wedge \frac{\tilde{\pi}(z^{\text{prop}}) w(z^{\text{prop}}, z^{\text{curr}}) \sum_{z \in \mathcal{S}_{a:b}(z^{\text{curr}})} w(z^{\text{curr}}, z)}{\tilde{\pi}(z^{\text{curr}}) w(z^{\text{curr}}, z^{\text{prop}}) \sum_{z \in \mathcal{S}_{a:b}(z^{\text{prop}})} w(z^{\text{prop}}, z)},$$

which is

$$\frac{1(c \in \{0,\ldots,K\}) 1(z^{\text{prop}} \in \mathcal{S}_{a:b}(z^{\text{curr}}))}{K+1} \times \left\{ \frac{\tilde{\pi}(z^{\text{curr}}) w(z^{\text{curr}}, z^{\text{prop}})}{\sum_{z \in \mathcal{S}_{a:b}(z^{\text{curr}})} w(z^{\text{curr}}, z)} \wedge \frac{\tilde{\pi}(z^{\text{prop}}) w(z^{\text{prop}}, z^{\text{curr}})}{\sum_{z \in \mathcal{S}_{a':b'}(z^{\text{prop}})} w(z^{\text{prop}}, z)} \right\}.$$

This expression is invariant to $(z^{\text{curr}}, a, b, c) \leftrightarrow (z^{\text{prop}}, a', b', c')$. □

Remark 1. The AAPS could be applied with a numerical integration scheme that does not have a Jacobian of 1. In such a scheme, however, the Jacobian would need to be included in the acceptance probability; moreover, a Jacobian other than 1 would lead to greater variability in $\tilde{\pi}$ along a path and, we conjecture, to a reduced efficiency.

The path $\mathcal{S}_{a:b}$ may visit parts of the statespace where the numerical solution to (2) is unstable and the error in the Hamiltonian may increase without bound, leading to wasted computational effort as large chunks of the path may be very unlikely to be proposed and accepted. Indeed it is even possible for the error to increase beyond machine precision. The no U-turn sampler suffers from a similar problem and we introduce a similar stability condition to that in Hoffman and Gelman (2014). We require that

$$\max_{z \in \mathcal{S}_{a:b}(z^{\text{curr}})} H(z) - \min_{z \in \mathcal{S}_{a:b}(z^{\text{curr}})} H(z) < \Delta, \qquad (7)$$

for some prespecified parameter Δ. This criterion can be monitored as the forward and backward integrations proceed, and if at any point the criterion is breached, the path is thrown away: the proposal is automatically rejected. Segment invariance means that $z \in \mathcal{S}_{a:b}(z^{\text{curr}}) \Leftrightarrow z \in \mathcal{S}_{a':b'}(z^{\text{prop}})$, so the same rejection would have occured if we created the path from any proposal in $\mathcal{S}_{a:b}(z^{\text{curr}})$, and detailed balance is still satisfied. For the experiments in Section 4 we found that a value of Δ = 1000 was sufficiently large that the criterion only interfered when something was seriously wrong with the integration. Step 3 of the algorithm then becomes:

1. Create $\mathcal{S}_{a:b}$ by leapfrog stepping <u>forward</u> from (x_0, p_0) and then <u>backward</u> from (x_0, p_0). If condition (7) fails then go to 6.

For a *d*-dimensional Markov chain, $\{X_t\}_{t=0}^{\infty}$, with a stationary distribution of *π*, the asymptotic variance is $V_f := \lim_{n\uparrow\infty} n\text{Var}\left[\frac{1}{n}\sum_{i=1}^{n} f(X_i)\right]$. Thus, after burn in, $\text{Var}\left[\frac{1}{n}\sum_{i=1}^{n} f(X_i)\right] \approx V_f / n$. The effective sample size is the number of iid samples from *π* that would lead to the same variance: $\text{ESS}_f = n\text{Var}_\pi[f]/V_f$. Let ESS_i be an empirical estimate of the ESS for the *i*th component of *X* (*i.e.*, *f* gives the *ith* component) and let n_{leap} be the total number of leapfrog steps taken during the run of the algorithm. We measure the efficiency of an algorithm as:

$$\text{Eff} = \frac{1}{n_{leap}} \min_{i=1,\dots,d} \text{ESS}_i. \quad (8)$$

Since the leapfrog step is by far the most computationally intensive part of the algorithm, Eff is proportional to the number of effective iid samples generated per second in the worst mixing component.

3.3 Choice of weight function

The weight function $w(z,z') = \tilde{\pi}(z')$ mentioned previously is a natural choice, and substitution into (6) shows that this leads to an acceptance probability of 1; however, it turns out not to be the most efficient choice. For example, intuitively the algorithm might be more efficient if points on the path that are further away from the current point are more likely to be proposed. To investigate the effects of the choice of w we examine six possibilities:

1. $w(z,z') = \tilde{\pi}(z')$, which leads to $\alpha = 1$.
2. $w(z,z') = \| x' - x \|^2$, squared jumping distance (SJD).
3. $w(z,z') = \tilde{\pi}(z') \| x' - x \|^2$, SJD modulated by target.
4. $w(z,z') = \| x' - x \|$, absolute jumping distance (AJD).
5. $w(z,z') = \tilde{\pi}(z') \| x' - x \|$, AJD modulated by target.
6. $w(z,z') = \tilde{\pi}(z') 1(z' \in \mathcal{H}(z))$, where $\mathcal{H}(z)$ is described below; this also leads to α = 1.

Scheme 6 essentially partitions $\mathcal{S}_{a:b}(z^{\text{curr}})$ into two halves of roughly equal total $\tilde{\pi}$ and then proposes only values from the half that does not contain $z^{\text{curr}}, \mathcal{H}(z^{\text{curr}})$; details are given in Appendix H.

The left panel of Figure 3 shows, for a particular choice of ϵ and target, the efficiency of AAPS as a function of K for each of the six weight schemes. Scheme 6 is the most efficient; however Schemes 3 and 5 each of which involve some measure of jumping distance modulated by $\tilde{\pi}$ are not far behind. Indeed there is only a factor of two between the least and most efficient. Very similar relative performances were found for all the other toy targets from Section 4.1 and across a variety of choices of ϵ, except that when ϵ becomes small, modulation of SJD or AJD by $\tilde{\pi}$ makes little difference since relative changes in $\tilde{\pi}$ are small.

A simple heuristic can explain the difference of a factor of nearly 2 between Scheme 1 and Scheme 6 in the case where $\tilde{\pi}$ is approximately constant; for example, when ϵ is small. $\mathcal{S}_{a:b}$ is constructed prior to making the proposal, so the computational effort does not depend on the scheme; thus efficiency can be measured crudely in terms of the squared jumping distance of the proposal (e.g. Roberts and Rosenthal, 2001; Sherlock and Roberts, 2009; Beskos et al., 2013), since it is accepted with probability 1. Without loss of generality, we rescale $\mathcal{S}_{a:b}$ to have unit length. For two independent Unif$(0,1)$ variables U_1 and U_2, Scheme 1 is equivalent to an expected squared jumping distance (ESDJ) of $\mathbb{E}\left[(U_1 - U_2)^2\right] = 1/6$, whereas Scheme 2 is equivalent to an ESJD of $\mathbb{E}\left[\{(1-U_1/2) - U_2/2)\}^2\right] = 7/24$; the ratio between the two ESJDs is 7/4.

A naive implementation of each of the above weighting schemes would require storing each of the points in $\mathcal{S}_{a:b}(z^{\text{curr}})$, which has an associated memory cost of $\mathcal{O}(K)$ and an exact cost that varies from one iteration to the next as the number of points in each segment is not fixed. For all schemes except the last there is a simple mechanism for sampling z^{prop} with a fixed $\mathcal{O}(1)$ memory cost; however, this would be useless if calculation of the acceptance ratio α in (6) still required storage of $\mathcal{O}(K)$. For Schemes 1, 2 and 3, however, it is also possible to calculate α with a fixed $\mathcal{O}(1)$ memory usage. We have found that this has a negligible effect on the CPU cost but the impact on the peak memory footprint of the code is substantial, decreasing by a factor of around 17 in d = 40, 27 in d = 100 and 42 in d = 800. Since there is little otherwise to choose between the schemes, we opt for the most efficient of these, Scheme 3, $w(z, z') = \| x' - x \|^2 \, \tilde{\pi}(x')$ and *apply this thoughout the remainder of this article*. Appendix B details the implementation of Schemes 1-3 with $\mathcal{O}(1)$ memory cost.

3.4 Robustness and efficiency

The robustness of AAPS with Scheme 3 which is visible in Figure 1 and similar figures in Section 4.1, arises from two aspects of the algorithm. We first consider robustness to the choice of ϵ, then robustness to the choice of K.

For Scheme 2, as with HMC itself, the acceptance probability contains the ratio $\tilde{\pi}(z^{\text{prop}}) / \tilde{\pi}(z^{\text{curr}})$. The error in the total energy of HMC is proportional to ϵ^2, so as ϵ increases the acceptance rate drops substantially when the error becomes $\mathcal{O}(1)$. By contrast, the acceptance probability for Scheme 3 contains only weighted sums of the $\tilde{\pi}$ in both the numerator and denominator so the empirical acceptance rate is much more robust to increasing ϵ. This effect can be seen in the right panel of Figure 3.

Robustness to the choice of K arises because AAPS samples a point from the whole set of segments rather than choosing the end point of the integration as HMC does. Figure 2 (bottom left) of Appendix D echoes Figure 1, but for an alternative version of AAPS which always sets $c = 0$ and always samples the proposal from segment K via weighting Scheme 3. At its optimum, the algorithm is slightly more efficient than the version of AAPS that we recommend; however, the efficiency drops much more sharply when K is larger or smaller than its optimal value.

Both of the above robustness properties are shared by Scheme 1 (also demonstrated in Figure 2); however, Scheme 3 is more efficient than Scheme 1 precisely because it preferentially targets proposals that are further from the current value.

3.5 Tuning

Since AAPS with weighting Scheme 3 is much more robust to the choice of either tuning parameter than HMC is, precise tuning is less important than it is for HMC. Thus, we present only brief guidelines here; further heuristics and empirical evidence for them are presented in Appendices E and F.

For any given $K > 0$, we recommend increasing ϵ from some small value until the empirical acceptance starts to change, stopping when the change from the small-ϵ acceptance rate is no more than 3%. Empirically, we have found that such minor changes in acceptance rate correspond to substantial changes in the total energy over the $K + 1$ segments, such that the modulation of SJD by $\tilde{\pi}$ starts to have a negative impact on the choice of z^{prop}.

For a given value of ϵ we recommend a short tuning run of AAPS using a large value, K_*, of K and then choosing a sensible $K \in \{0, \dots, K_*\}$ according to the most popularly proposed segment number using a diagnostic that we describe in Appendix F.

As with HMC and the no U-turn sampler, the mass matrix, M, used by AAPS can also be tuned, and mixing will be optimised if $M^{-1} \approx \text{Var}_\pi[X]$, as approximated from a tuning run. However, the matrix-vector multiplication required for simulating p_0 and in each leapfrog step can be expensive, so it is usual to choose a diagonal *M* instead.

3.6 Gaussian process limit

Intuitively, the more eccentric a target the more apogees one might expect per unit time. This section culminates in an expression which makes this relationship concrete for a product target where each component has its own length scale. With the initial condition (X_0, P_0) sampled from $\tilde{\pi}, P_t^\top M^{-1}\nabla U(X_t)$ can be considered as a random function of time. We first show that when the true Hamiltonian dynamics are used, subject to conditions, a rescaling of this dot-product tends to a stationary Gaussian process as the number of components in the product tends to infinity. A formula for the expected number of apogees per unit time follows as a corollary.

We consider a *d*-dimensional product target with a potential of

$$U^{(d)}(x) = \sum_{i=1}^{d} g\left(\sqrt{\nu_i^{(d)}} x_i^{(d)}\right) + \text{constant} \quad (9)$$

for some $g: \mathbb{R} \to \mathbb{R}$ and values $\nu_i^{(d)} > 0, i = 1, \ldots, d$, and where $\int \exp\{-U^{(d)}(x)\}\mathrm{d}x = 1$.

HMC using a diagonal mass matrix, *M*, and a product target is equivalent to HMC using an identity mass matrix and a target of $M^{-1/2}x$ (e.g. Neal, 2011b), which, in the case of (9) is also a product target, but with different ν_i. For simplicity, therefore, we assume the identity mass matrix throughout this section. We also consider the true Hamiltonian dynamics which are approached in the limit as $\epsilon \downarrow 0$, but approximate the dynamics for small to moderate ϵ reasonably well.

Define the scaled dot product at time *t* given an initial position of $x_0^{(d)}$ and momentum of $p_0^{(d)}$ as

$$D^{(d)}(t; x_0^{(d)}, p_0^{(d)}) := \frac{1}{\sqrt{d}} p^{(d)}(t) \cdot \nabla_x U|_{x^{(d)}(t)} = \frac{1}{\sqrt{d}} \sum_{i=1}^{d} \sqrt{\nu_i^{(d)}} g'\left(\sqrt{\nu_i^{(d)}} x_i^{(d)}\right) p_i^{(d)}(t). \quad (10)$$

We define the one-dimensional densities with respect to Lebesgue measure

$$\pi_\nu(x) = \sqrt{\nu}\exp\{-g(\sqrt{\nu}x)\} \text{ and } \rho_1(p) = \frac{1}{\sqrt{2\pi}}\exp\{-p^2/2\},$$

and let π_ν and ρ_1 denote the corresponding measures. The joint density and measure are $\tilde{\pi}_\nu(x,p;\nu) = \pi_\nu(x;\nu)\rho_1(p)$ and $\tilde{\pi}_\nu$.

Assumptions 1. We assume that $g \in C^1$, that there is a $\delta > 0$ such that

$$\mathbb{E}_{X\sim\pi_1}\left[|g'(X)|^{2+\delta}\right] < \infty, \qquad (11)$$

and that for each $y_0 \in \mathbb{R}$, there is a unique, non-explosive solution $a(t; y_0, y'_0)$ to the initial value problem:

$$\frac{\mathrm{d}^2 y}{\mathrm{d}t^2} = -g'(y); y(0) = y_0, y'(0) = y'_0. \quad (12)$$

Theorem 1 is proved in Appendix C.1.

Theorem 1. *Let the potential be defined as in (9) and where g satisfies the assumptions around (11) and (12). Further, let μ be a distribution with support on* $\mathbb{R}^+$ *with*

$$\mathbb{E}_{\nu\sim\mu}\left[\nu^{1+\delta/2}\right] < \infty \qquad (13)$$

for some $\delta > 0$*, and let* $\nu_i \overset{iid}{\sim} \mu$*. Define*

$$V(t) = \mathbb{E}\left[\nu g'(X) P g'(\sqrt{\nu}X_t(X,P)) a'(\sqrt{\nu}t; X, P)\right], \quad (14)$$

where the expectation is over the independent variables $X \sim \pi_1, P \sim \rho_1$ *and* $\nu \sim \mu$*, and assume that for any finite sequence of n distinct times* $(t_1, \ldots, t_n)$*, the n × n matrix Σ with* $\Sigma_{i,j} = V(t_j - t_i)$ *is positive definite.*

Let $D^{(d)}$ *be the scaled dot product defined in (10), and let* $(X_0, P_0) \sim \tilde{\pi}$*; then*

$$D^{(d)} \Rightarrow \tilde{D} \sim \mathrm{SGP}(0, V), \qquad (15)$$

as $d \to \infty$, *where* $Y \sim \text{SGP}(b, V)$ *denotes that Y is a one-dimensional stationary Gaussian process with an expectation of b and a covariance function of* $\text{Cov}\left[Y_t, Y_{t'}\right] = V(t' - t)$.

Ylvisaker (1965) shows that the expected number of zeros over a unit interval of a stationary Gaussian process with a unit variance is: $\frac{1}{\pi}\sqrt{-C''(0)}$, where *C* is its covariance function. Zeroes can be either apogees or perigees (local minima in *U*(*x*) along the path), and these alternate. Hence, with some work (see Appendix C.2), we obtain the following:

Corollary 1. *The expected number of apogees of* $\tilde{D}$ *over a time T is*

$$\mathbb{E}\left[N(T)\right] \propto T\sqrt{\mathbb{E}\left[\nu\right]} \times \sqrt{\frac{\mathbb{E}\left[\nu^2\right]}{\mathbb{E}\left[\nu\right]^2}}.$$

Since $P_0 \sim N(0, I_d)$, with an identity mass matrix, the root-mean-square speed in any direction is 1. As *ν* is a squared inverse-scale parameter, the first term in the product simply relates the time interval to the overall length scale of the target ($1/\sqrt{\mathbb{E}\left[\nu\right]}$). The second part of the product increases with *relative* variability in the squared inverse length scales of the components of the target. Choosing *K* fixes *N*(*T*), and approximately fixes the product of the two terms on the right. For a given *relative* variability in length scales, the integration time automatically flexes according to the overall absolute length scale; however if that relative variability increases then *T* and, hence, the path length reduces. This makes it plain that the tuning parameter *K* relates to properties *intrinsic* to the target; unlike *L*, its impact is relatively unaffected by a uniform redefinition of the length scale of the target or by the choice of *ϵ*.

4 Numerical Experiments

In this section we compare, AAPS with HMC, blurred HMC (see Section 1) and the no U-turn sampler over a variety of targets. For fairness we use the basic no U-turn sampler from Hoffman and Gelman (2014), without it needing to adaptatively tune *ϵ*; instead tuning *ϵ* using a fine grid of values. For both varieties of HMC we use a grid of ϵ and *L* values, and for AAPS we use a grid of ϵ and *K* values; in each case we choose the combination that leads to the optimal efficiency. To ensure that the various toy targets are close to representing the difficulties encountered with targets from complex statistical models, all algorithms use the identity mass matrix.

All code (AAPS/HMC/no U-turn sampler) was written in `C++`; the effective sample size was estimated using the `R` package `coda`, and the mean number of leapfrog steps per iteration (for AAPS and the no U-turn sampler) was output from each run as a diagnostic.

4.1 Toy targets

Here we investigate performance across a variety of targets with a relatively simple functional form, and across a variety of dimensions. Many are product densities with independent components: Gaussian, logistic and skew-Gaussian. We consider different, relatively large ratios ξ between the largest and smallest length scales of the components in a target, as well as different sequences of scales from the smallest to the largest. We also consider a modification of the Rosenbrock banana-shaped target.

For a target of dimension *d*, given a sequence of scale parameters, $\sigma_1, \ldots, \sigma_d$ we consider the following four forms:

$$\pi_G(x) = \prod_{i=1}^{d} \frac{1}{\sigma_i\sqrt{2\pi}} \exp\left(-\frac{x_i^2}{2\sigma_i^2}\right),$$

$$\pi_L(x) = \prod_{i=1}^{d} \frac{1}{\sigma_i} \frac{\exp(x_i / \sigma_i)}{\{1+\exp(x_i / \sigma_i)\}^2},$$

$$\pi_{SG}(x) = \prod_{i=1}^{d} \frac{2}{\sigma_i\sqrt{2\pi}} \exp\left(-\frac{x_i^2}{2\sigma_i^2}\right) \Phi\left(\frac{\alpha x_i}{\sigma_i}\right),$$

$$\pi_{MR}(x) \propto \prod_{i=1}^{d/2} \exp\left\{-\frac{1}{2s_i^2}(x_{2i-i} - \sqrt{2}\beta s_i)^2\right\} \exp\left\{-\frac{1}{2}\left(x_{2i} - \frac{1}{\sqrt{2}s_i} \frac{x_{2i-1}^2}{1+x_{2i-i}^2/(4s_i^2)}\right)^2\right\},$$

where Φ is the distribution function of a $\mathrm{N}(0,1)$ variable, α = 3, β = 1 and $s_i^2 = 99(i-1)/(d/2-1)+1$. The targets π_G, π_L an π_{SG} are products of one-dimensional Gaussian, logistic and skew-Gaussian distributions respectively. The potential surface of the target π_{MR} is a modified version of the Rosenbrock function (Rosenbrock, 1960), a well-known, difficult target for optimisation algorithms, which is also a challenging target used to benchmark MCMC algorithms (e.g. Pagani et al., 2021; Heng and Jacob, 2019). The tails of the standard Rosenbrock potential increase quartically, but all algorithms which use a standard leapfrog step and Gaussian momentum are unstable in the tails of a target where the potential increases faster than quadratically. We have, therefore, modified the standard Rosenbrock target to keep the difficult banana shape whilst ensuring quadratic tails.

For π_G, π_L and π_{SG}, we denote the largest ratio of length scales by $\xi := \max_{1\le i \le j \le d} \sigma_i / \sigma_j$ and define four different patterns between the lowest and highest scaling, depending on which of σ_i, $\sigma_i^2, 1/\sigma_i^2$ or $1/\sigma_i$ increases approximately linearly with component number. To minimise the odd behaviour that HMC (but not AAPS or the no U-turn sampler) can exhibit when scalings are rational multiples of each other (a phenomenon which is rarely seen for targets in practice) we jitter the scales for all the intermediate components. Specifically, let $\underline{v}$ be a vector with $v_1 = 0$, v_d = 1 and for $i = 2,\dots,d-1, v_i = (i-1+U_i)/(d-1)$ where $U_2,\dots,U_{d-1}$ are independent $\text{Unif}(-0.5, 0.5)$ variables. Then we define the following four progressions for $i = 1,\dots,d$: SD: $\sigma_i = (\xi - 1)v_i + 1$; VAR: $\sigma_i^2 = (\xi^2 - 1)v_i + 1$; H: $1/\sigma_i^2 = (1 - 1/\xi^2)v_i + 1/\xi^2$; invSD: $1/\sigma_i = (1 - 1/\xi)v_i + 1/\xi$.

A final target, π_G^{RN}, arises from an online comparison between HMC and the no U-turn sampler at

https://radfordneal.wordpress.com/2012/01/27/evaluation-of-nuts-more-comments-on-the-paper-by-hoffman-and-gelman/.

Figure 4 uses ξ = 20 and repeats Figure 1 in d = 40 for each of the main product target types: Gaussian, skew-Gaussian and logistic. It demonstrate the robustness of AAPS when compared with HMC and blurred HMC. Appendix D contains more details on these experiments. Figure 1 in Appendix D plots efficiency against step size when the popular No U-turn Sampler is used on the targets in Figures 1 and 4. The peaks for the logistic and modified Rosenbrock targets are broad, suggesting some robustness; however, the peaks for the Gaussian and skew-Gaussian targets are much sharper.

We next investigate products of Gaussians using each of the four scale-parameter progressions with $\xi = 20$. Dimension $d = 40$ is high enough that for this and higher dimensions the components can be well approximated as arising from some continuous distributions and Corollary 1 applies.

Table 1 shows the efficiencies of HMC, blurred HMC and the no U-turn sampler relative to that of AAPS. Amongst the Gaussian targets, the relative performance of AAPS compared with HMC and NUTS is best for π_G^{invSD} and π_G^{H}, which have just a few components with large scales and many components with small scales; weighting Scheme 3 ensures that the large components are explored preferentially. In contrast, the large number of components with large scales in π_G^{VAR} leads to the worst relative performance of AAPS; we, therefore, investigate this regime further using alternative component distributions, and choice of dimension and ξ. Across the range of targets, no algorithm is more than 1.7 times as efficient as AAPS. Empirical acceptance rates at the optimal parameter settings are provided in Appendix E. All ESS estimates were at least 1000.

Table 3 of Appendix 1 provides the equivalent efficiencies when the algorithms are tuned according to recommended guidelines, and shows AAPS remaining competitive with blurred HMC and the no U-turn sampler.

4.2 Stochastic volatility model

Consider the following model for zero-centred, Gaussian data $y = (y_1, \ldots, y_T)$ where the variance depends on a zero-mean, Gaussian AR(1) process started from stationarity (e.g. Girolami and Calderhead, 2011; Wu et al., 2019):

$$Y_t \sim \mathrm{N}(0, \kappa^2 \exp x_t), t = 1, \ldots, T,$$

$$X_1 \sim \mathrm{N}\left(0, \frac{\sigma^2}{1-\phi^2}\right), \quad X_t \mid (X_{t-1} = x_{t-1}) \sim \mathrm{N}(\phi x_{t-1}, \sigma^2), t = 2, \ldots, T.$$

Parameter priors are $\pi_0(\kappa) \propto 1/\kappa, 1/\sigma^2 \sim \text{Gamma}(10, 0.05)$ and $(1+\phi)/2 \sim \text{Beta}(20, 1.5)$, and we reparameterise to give a parameter vector in $\mathbb{R}^3$:

$$\alpha = \log \frac{1+\phi}{1-\phi}, \ \beta = \log \kappa, \ \text{and} \ \gamma = 2 \log \sigma.$$

As in Girolami and Calderhead (2011) and Wu et al. (2019) we generate *T* = 1000 observations using parameters $\phi = 0.98, \kappa = 0.65$ and $\sigma = 0.15$. We then apply blurred HMC, AAPS and the no U-turn sampler to perform inference on the 1003-dimensional posterior. We ran AAPS using two different *K* values, one found by optimising the choice of (ϵ, K) over a numerical grid and one by using the tuning mechanism mentioned in Section 3.5. Tuning standard HMC (not blurred HMC) on such a high-dimensional target was extremely difficult; due to the algorithm's sensitivity to the integration time, we could not identify a suitable range for *L*. Widely used statistical packages such as Stan (Stan Development Team, 2020) and PyMC3 (Salvatier et al., 2016) perform the blurring by default, and so we only present the results for HMC-bl.

Each algorithm was then run for ten replicates of 10^5 iterations using the optimal tuning parameters. Efficiency was calculated for each parameter, and the minimum efficiency over the latent variables and over all 1003 components were also calculated. For each algorithm the mean and standard deviation (over the replicates) of these efficiencies were ascertained; Table 2 reports these values normalised by the mean efficiency for AAPS for that parameter or parameter combination. Overall, on this complex, high-dimensional posterior, AAPS is slightly less efficient than blurred HMC, and slightly more efficient than the no U-turn sampler.

4.3 Multimodality

The AAPS algorithm with *K* = 0 is close to reducible on a multimodal one-dimensional target, and this might lead to concerns about the algorithm's performance on multimodal targets in general. However, in *d* dimensions, because $p^\top \nabla U(x)$ is a sum of *d* components even with *K* = 0, AAPS is not reducible on multimodal targets with *d* > 1. This is illustrated in Appendix G, which also details a short simulation study on three 40-dimensional bimodal targets where AAPS is more efficient than the no U-turn sampler and is never less than two-thirds as efficient as blurred HMC.

5 Discussion

We have presented the *Apogee to Apogee Path Sampler* (AAPS), and demonstrated empirically that it has a similar efficiency to HMC but is much easier to tune. From a current point, AAPS uses the leapfrog step to create a path consisting of a fixed number of *segments*, it then proposes a point from these segments and uses an accept-reject step to ensure that it targets the intended distribution.

We investigated six possible mechanisms for proposing a point from the path, and for the numerical experiments we chose the probability of proposing a point to be proportional to the product of the extended target density at that point and the proposal's squared distance from the current point, which was possible with an $\mathcal{O}(1)$ memory cost. However, the flexibility in the proposal mechanism allows other possibilities such as a Mahalanobis distance based on an estimated covariance matrix, or $(\| x^{\text{prop}} - \mu \|^2 - \| x^{\text{curr}} - \mu \|^2)^2$ for some central point, μ, with a similar motivation to the ChEEs diagnostic of Hoffman et al. (2021). Indeed, if any scalar or vector function, f, is of particular interest, then a proposal weighting of the form $\| f(x^{\text{prop}}) - f(x^{\text{curr}}) \|^2$ could be used with a memory cost of $\mathcal{O}(1)$.

Choosing the current segment's position uniformly at random from the K + 1 segments is not the only way to preserve detailed balance with respect to the intended target. For example, the current segment could be fixed as segment 0 and proposals could only be made from segment K, a choice which bears some resemblance to the window scheme in Neal (1992); however, we found that this had a negative impact on the robustness of the efficiency to the choice of K (see Appendix D).

Because of its simplicity, many extensions to the AAPS algorithm are clear. For example, if the positions along the path are stored, then a delayed rejection step may increase the acceptance probabilities. A cheap surrogate for π could be substituted within $\tilde{\pi}$ in any weighting scheme. Indeed, given c, the randomly chosen offset of segment 0, the next value could be chosen conditional on z^{curr} using any Markov kernel reversible with respect to $\tilde{\pi}$ (see also Neal, 2011a). A non-reversible version of the algorithm could set $K = 1$ and always choose a path consisting of the current segment and the next segment forward; instead of completely refreshing momentum at each iteration, the momentum at the start of a new step could be a Crank-Nicolson perturbation of the momentum at the end of the previous step as in Horowitz (1991). The properties of the leapfrog step required for the validity of AAPS are a subset of those required for HMC (see Section 2.2), so any alternative momentum formulation (e.g. Livingstone et al., 2019) or numerical integration scheme that can be used within HMC could also be used within AAPS.

AAPS with weighting Scheme 1 relates to HMC using windows of states Neal (2011b) but with K defining the total number of (forward and backward) leapfrog steps taken rather than the number of additional segments. As mentioned in Section 1 and shown in Corollary 1, the number of apogees is a more natural tuning parameter than an integration time as it relates to intrinsic properties of the target: rescaling all co-ordinates by a constant factor would not change the optimal K.

Acknowledgements and Disclosure

Work by all authors was supported by EPSRC grant EP/P033075/1. We are grateful to two anonymous reviewers for comments and suggestions that have materially improved the article. There are no financial or non-financial conflicts of interest to declare.

Code is available from https://github.com/ChrisGSherlock/AAPS

Supplementary Material

Appendices A-I are contained in the supplementary material.

References

Beskos, A., Pillai, N., Roberts, G., Sanz-Serna, J.-M. and Stuart, A. (2013) Optimal tuning of the hybrid Monte Carlo algorithm. *Bernoulli*, **19**, 1501–1534. URL:http://www.jstor.org/stable/42919328.

Bou-Rabee, N. and Sanz-Serna, J. M. (2017) Randomized Hamiltonian Monte Carlo. *The Annals of Applied Probability*, **27**, 2159–2194. URL:http://www.jstor.org/stable/26361544.

Brooks, S., Gelman, A., Jones, G. L. and Meng, X.-L. (eds.) (2011) *Handbook of Markov chain Monte Carlo*. Chapman & Hall/CRC Handbooks of Modern Statistical Methods. Boca Raton, FL: CRC Press.

Duane, S., Kennedy, A., Pendleton, B. J. and Roweth, D. (1987) Hybrid Monte Carlo. *Physics Letters B*, **195**, 216–222. URL:https://www.sciencedirect.com/science/article/pii/037026938791197X.

Gilks, W. R., Richardson, S. and Spiegelhalter, D. J. (1996) *Markov Chain Monte Carlo in practice*. London, UK: Chapman and Hall.

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**, 123–214. URL:https://rss.onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-9868.2010.00765.x.

Heng, J. and Jacob, P. E. (2019) Unbiased Hamiltonian Monte Carlo with couplings. *Biometrika*, **106**, 287–302. URL:https://doi.org/10.1093/biomet/asy074.

Hoffman, M., Radul, A. and Sountsov, P. (2021) An adaptive-MCMC scheme for setting trajectory lengths in Hamiltonian Monte Carlo. In *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics* (eds. A. Banerjee and K. Fukumizu), vol. 130 of *Proceedings of Machine Learning Research*, 3907–3915. PMLR. URL:https://proceedings.mlr.press/v130/hoffman21a.html.

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**, 1593–1623.

Horowitz, A. M. (1991) A generalized guided Monte Carlo algorithm. *Physics Letters B*, **268**, 247–252. URL:https://www.sciencedirect.com/science/article/pii/0370269391908125.

Livingstone, S., Faulkner, M. F. and Roberts, G. O. (2019) Kinetic energy choice in Hamiltonian/hybrid Monte Carlo. *Biometrika*, **106**, 303–319. URL:https://doi.org/10.1093/biomet/asz013.

Mackenze, P. B. (1989) An improved hybrid Monte Carlo method. *Physics Letters B*, **226**, 369–371.

Neal, R. M. (1992) An improved acceptance procedure for the hybrid Monte Carlo algorithm. *Journal of Computational Physics*, **111**, 194–203.

— (2011a) MCMC using ensembles of states for problems with fast and slow variables such as Gaussian process regression.

— (2011b) MCMC using Hamiltonian dynamics. In *Handbook of Markov chain Monte Carlo* (eds. S. Brooks, A. Gelman, G. Jones and X.-L. Meng), chap. 5, 113–162. CRC press.

Pagani, F., Wiegand, M. and Nadarajah, S. (2021) An n-dimensional Rosenbrock distribution for Markov chain Monte Carlo testing. *Scandinavian Journal of Statistics*. URL:https://onlinelibrary.wiley.com/doi/abs/10.1111/sjos.12532.

Roberts, G. O. and Rosenthal, J. S. (2001) Optimal scaling for various Metropolis-Hastings algorithms. *Statistical Science*, **16**, 351–367.

Rosenbrock, H. H. (1960) An Automatic Method for Finding the Greatest or Least Value of a Function. *The Computer Journal*, **3**, 175–184. URL:https://doi.org/10.1093/comjnl/3.3.175.

Salvatier, J., Wiecki, T. V. and Fonnesbeck, C. (2016) Probabilistic programming in Python using PyMC3. *PeerJ Computer Science*, **2**, e55.

Sherlock, C. and Roberts, G. (2009) Optimal scaling of the random walk Metropolis on elliptically symmetric unimodal targets. *Bernoulli*, **15**, 774 – 798. URL:https://doi.org/10.3150/08-BEJ176.

Stan Development Team (2020) Stan modeling language users guide and reference manual. URL:http://mc-stan.org/. Version 2.28.

Wu, C., Stoehr, J. and Robert, C. P. (2019) Hamiltonian Monte Carlo by learning leapfrog scale.

Ylvisaker, N. D. (1965) The expected number of zeros of a stationary Gaussian process. *Annals of Mathematical Statistics*, **36**, 1043–1046.

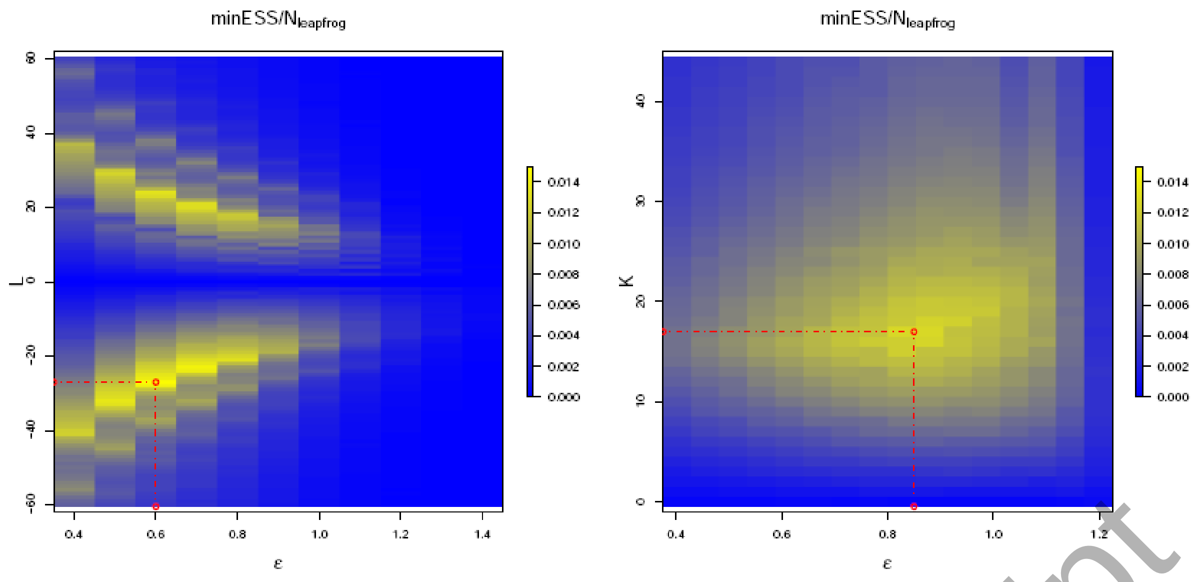

Fig. 1 Efficiency, according to (8), as a function of the tuning parameters for the 40-dimensional modified Rosenbrock target of Section 4.1. Left panel: HMC with positive *L* values corresponding to the standard HMC algorithm and negative *L* correspond to $|L|$ leapfrog steps of blurred HMC. Right panel: AAPS. Optimal parameter settings in red.

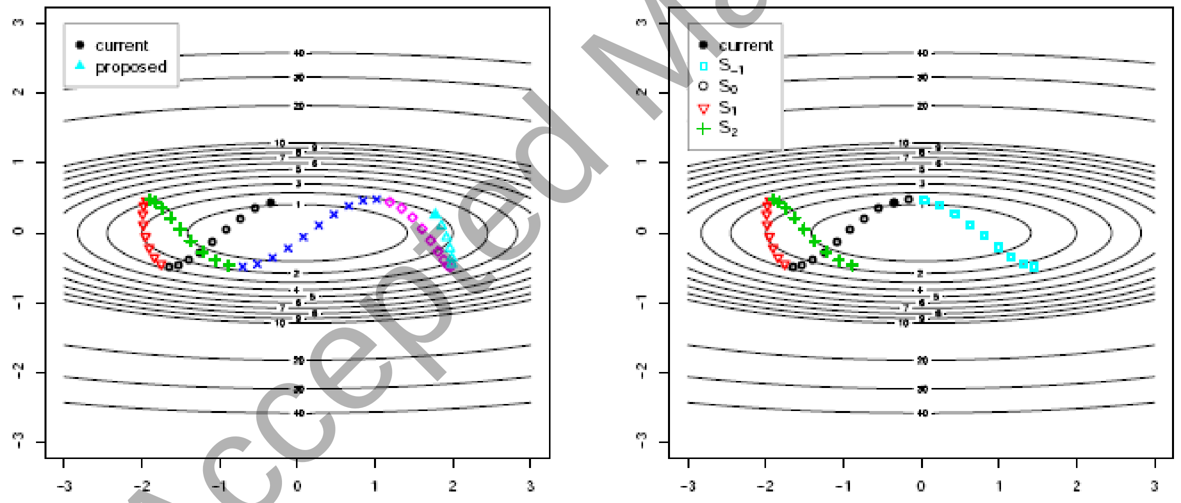

Fig. 2 Left panel: *L* = 50 leapfrog steps of size $\epsilon = 0.1$ from the current point. Right panel: the current segment, $\mathcal{S}_0$, and two segments forward and one segment backward using $\epsilon = 0.1$. The current point, x_0 is simulated from a target, which has a density of $\pi(x) \propto \exp(-x_1^2/2 - 6x_2^2)$; p_0 is simulated from $\mathrm{N}(0, I_2)$. Different colours and symbols are used for each segment along the path.

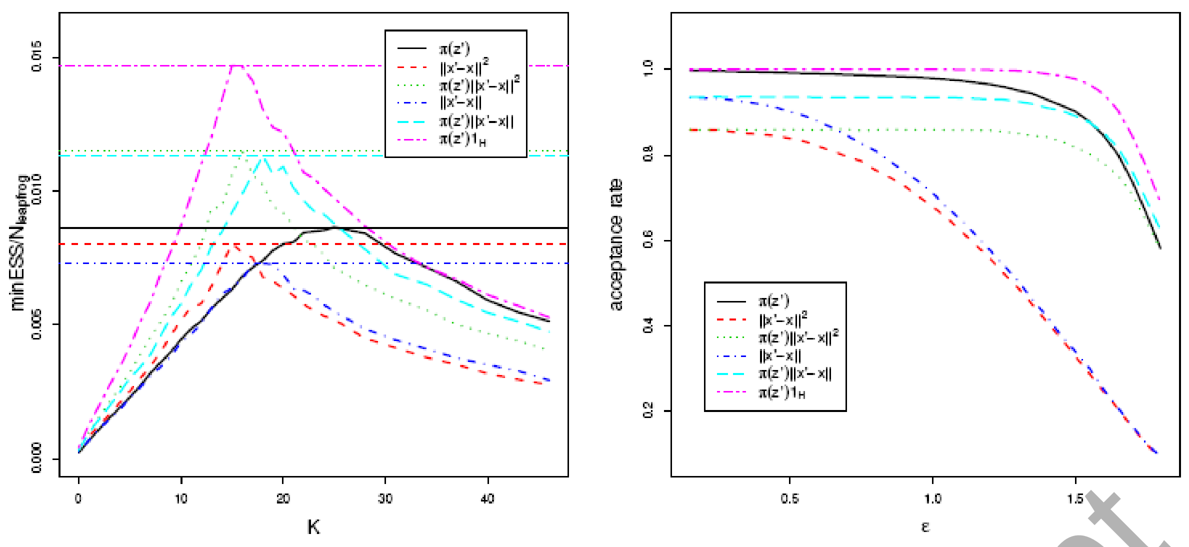

Fig. 3 Left: efficiency (see (8)) of AAPS as a function of K when $\epsilon = 1.2$. Right: acceptance rate as a function of ϵ when $K = 15$. There is one curve for each of the six choices of weight function. The single horizontal line associated with each curve in the left panel indicates the maximum efficiency achieved. The target is $\pi_G^H(x)$ (see Section 4.1) with $d = 40$ and $\xi = 20$.

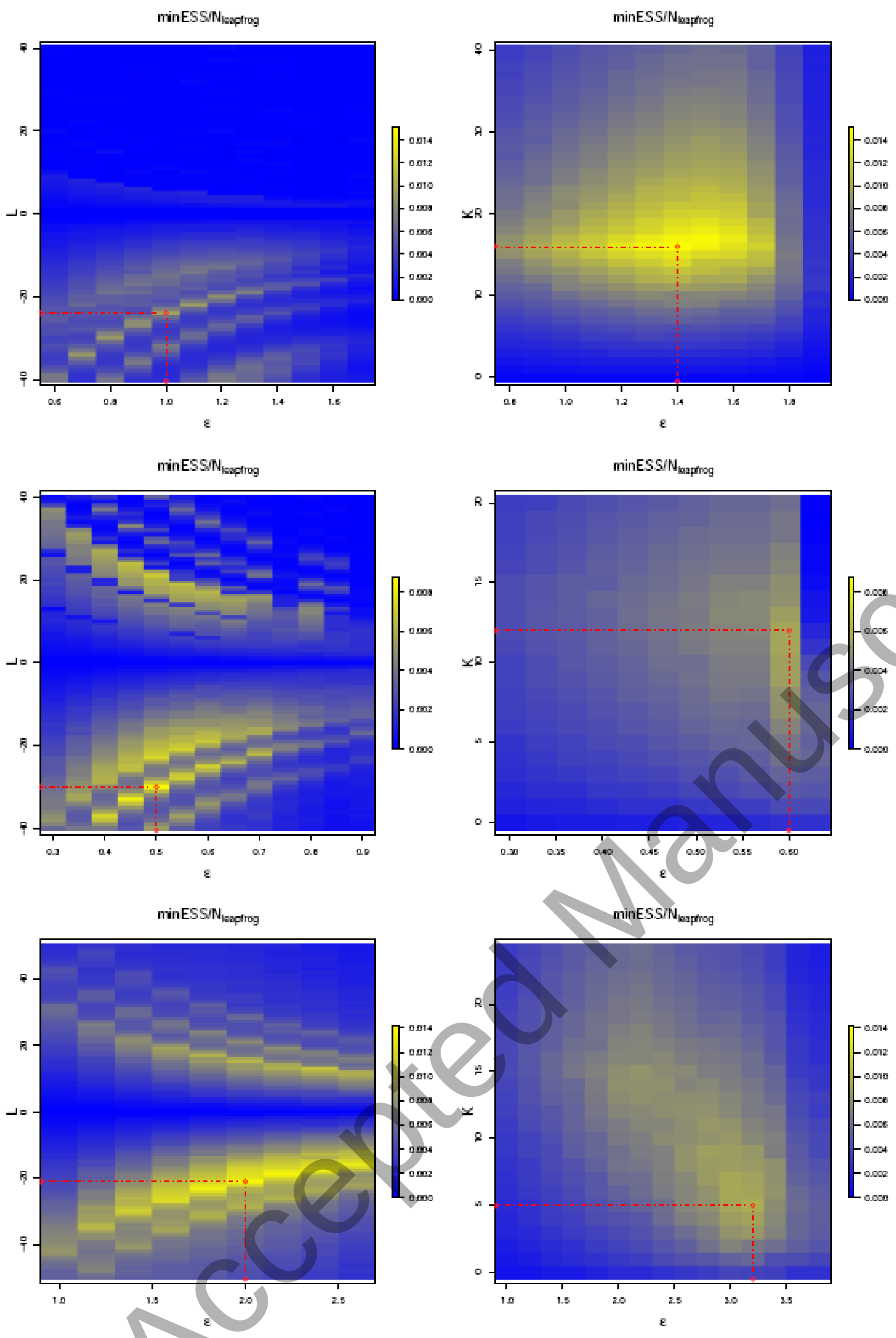

Fig. 4 Efficiency, measured via (8), as a function of the tuning parameters for π_G^H (top), π_{SG}^{VAR} (middle) and π_L^{VAR} (bottom) all with $(\xi, d) = (20, 40)$ as defined in Section 4.1. Left panels: HMC with positive *L* values corresponding to the standard HMC algorithm and negative *L* correspond to $|L|$ leapfrog steps of blurred HMC. Right panels: AAPS. Optimal tuning in red.

Table 1 Relative efficiency compared with AAPS of HMC, blurred HMC (HMC-bl) and the no U-turn sampler (NUTS); raw efficiencies were taken as the minimum over the *d* components of the number of effective samples per leapfrog step. The optimum was found by grid search. (1) This is not a typographic error: the values differ at the 5th decimal place. (2) ξ for odd-numbered components of π_{MR}. (3) The tuning surface for HMC was so uneven that were were unable to ascertain the optimal tuning parameters with any degree of certainty; instead we picked the least unreasonable of the combinations we tried.

Target type	d	ξ	AAPS	HMC	HMC-bl	NUTS
π_G^{SD}	40	20	1.000	0.722	0.718	1.182
π_G^{VAR}	40	20	1.000	1.016	1.091	1.461
π_G^{H}	40	20	1.000	(1) 0.162	0.644	0.392
π_G^{invSD}	40	20	1.000	(1) 0.162	0.461	0.460
π_{SG}^{VAR}	40	20	1.000	1.253	1.528	1.618
π_L^{VAR}	40	20	1.000	1.135	1.488	1.677
π_G^{VAR}	100	20	1.000	0.657	1.020	1.378
π_G^{VAR}	40	40	1.000	1.190	1.346	1.645
π_{MR}	20	(2) 10	1.000	1.647	1.582	0.728
π_{MR}	40	(2) 10	1.000	1.045	1.166	0.873
π_{MR}	100	(2) 10	1.000	0.770	0.970	1.079
π_{MR}	400	(2) 10	1.000	0.684	0.859	0.963
π_G^{RN}	30	110	1.000	(3) 0.019	1.206	0.306

Table 2 Relative efficiency compared with AAPS^g (AAPS tuned using a grid) of AAPS^a (AAPS tuned using the advice from Section 3.5), blurred HMC (HMC-bl) and the no U-turn sampler (NUTS) for the stochastic volatility model using 10 replicates of 10^5 iterations. Raw efficiencies were the number of effective samples per leapfrog step; these were then normalised by the mean efficiency from AAPS^g; normalised standard deviations are reported in brackets.

Parameter	AAPS^g	AAPS^a	HMC-bl	NUTS
A	1.00 (0.12)	1.04 (0.19)	1.05 (0.18)	0.73 (0.25)
B	1.00 (0.09)	0.87 (0.11)	1.04 (0.13)	1.24 (0.29)
Γ	1.00 (0.03)	1.20 (0.05)	1.14 (0.07)	0.74 (0.19)
$\min_{t\in\{1,\ldots,T\}} \mathrm{ESS}(X_t)$	1.00 (0.13)	0.89 (0.16)	1.07 (0.19)	0.78 (0.44)
$\min_{\alpha,\beta,\gamma,t\in\{1,\ldots,T\}} \mathrm{ESS}$	1.00 (0.05)	0.91 (0.12)	1.06 (0.11)	0.76 (0.21)
acc. rate (%)	75.1	75.5	66.5	86.9