Faster Hamiltonian Monte Carlo by Learning Leapfrog Scale

Changye Wu* · **Julien Stoehr*** · **Christian P. Robert**

Abstract Hamiltonian Monte Carlo samplers have become standard algorithms for MCMC implementations, as opposed to more basic versions, but they still require some amount of tuning and calibration. Exploiting the U-turn criterion of the NUTS algorithm (Hoffman and Gelman, 2014), we propose a version of HMC that relies on the distribution of the integration time of the associated leapfrog integrator. Using in addition the primal-dual averaging method for tuning the step size of the integrator, we achieve an essentially calibration free version of HMC. When compared with the original NUTS on several benchmarks, this algorithm exhibits a significantly improved efficiency.

Keywords Acceleration methods · No-U-Turn Sampler

1 Introduction

Hamiltonian Monte Carlo (HMC, Duane et al., 1987; Neal, 2011) has emerged as an efficient Markov Chain Monte Carlo (MCMC) sampling method, which, in particular, is able to deal with high-dimensional target distributions. The method relies on a deterministic differential flow stemming from Hamiltonian mechanics to produce transitions across the parameter space of an augmented distribution, referred to as Hamiltonian. Such time-continuous dynamics leave the augmented distribution invariant but in practice the latter need to be discretised. The transitions are hence computed using a gradient-based symplectic integrator – the most popular one being the second order Störmer-Verlet also known as leapfrog integrator (Hairer et al., 2003) – and a Metropolis-Hastings acceptance ratio need to be added towards correcting the approximate numerical solution and hence preserving the proper target measure.

C. Wu, J. Stoehr, C. P. Robert,
Université Paris-Dauphine, Université PSL, CNRS, CEREMADE, 75016 PARIS, FRANCE
E-mail: wu,stoehr,xian@ceremade.dauphine.fr

C. P. Robert
Department of Statistics, University of Warwick, UK
E-mail: stsmaz@warwick.ac.uk

* *These authors contributed equally to this work.*

Whilst the algorithm theoretically benefits from a fast exploration of the parameter space by accepting large transitions with high probability, this efficiency is marred by its high sensitivity to hand-tuned parameters, namely the step size ϵ of the discretisation scheme, the number of steps L of the integrator, and the covariance matrix M of the auxiliary variables. Indeed, the discretisation error due to a poorly tuned step size often leads to a low acceptance rate, since the variation in the Hamiltonians from leapfrog start to leapfrog finish is governed by ϵ^2. Furthermore, calibrating the number of steps of the integrator is paramount to avoid slow mixing chains or to save some computation cost. More specifically, if L is not large enough, HMC exhibits a random walk behaviour similar to standard Metropolis Hastings algorithms and may thus struggle to properly explore the support of the target. If L is too large the associated dynamic may retrace its steps back to a neighbourhood of the initial state (Betancourt et al., 2014) wasting computation efforts and diminishing mixing. Finally the choice of a well-suited covariance matrix for the auxiliary variable can enhance both the speed and the mixing of HMC. A recent proposal in this regard is the Riemann manifold approach of Girolami and Calderhead (2011) that adapts the covariance matrix of the auxiliary variables to the curvature of the target distribution, followed by the automatic tuning strategy of Wang et al. (2013).

Our focus in this paper is mostly connected with the number L. In this regard, the popular No-U-Turn Sampler (NUTS, Hoffman and Gelman, 2014) automatically tunes both the step size and the number of integration steps. This algorithm achieves at least the same efficiency as the standard HMC, when efficiency is defined through ESS and expected square jump distance (ESJD), albeit its implementation involves an extra cost that may prove burdensome. The core idea behind NUTS is to pick the step size via primal-dual averaging (Nesterov, 2009) in a burn-in phase and to build at each iteration a proposal based on a locally longest path on a level set of the Hamiltonian. This is achieved by a recursive algorithm that follows a level set to a turning point, doubling that path for time-reversibility reasons, and samples a point along that path with weights proportional to the target density. As a result, the algorithm produces a path that is much longer than the one needed for the resulting output. Here, we aim at reducing the computing cost induced by this feature of NUTS and we explore the opportunity to learn the empirical distribution of the longest NUTS path – consistent with the target distribution – by exploiting the U-turn criterion of NUTS for the level set visited during a burn-in phase. This empirical distribution is then used to randomly pick L at each iteration of a regular HMC scheme.

The paper begins with a brief description of HMC and NUTS in Section 2. We then explain how to learn the empirical distribution of L in Section 3. We illustrate the efficiency of the proposed algorithm compared to the efficient NUTS implemented by `Rstan` (Stan Development Team, 2018a) on several benchmark examples in Section 4, and conclude in Section 5.

2 Hamiltonian Monte Carlo

We consider a probability target measure π on $\Theta \subset \mathbb{R}^d$ with density, also denoted π, against the Lebesgue measure, written as $\pi(\theta) \propto \exp\{-U(\theta)\}$, where U is a continuously differentiable function called the potential function. HMC (Neal, 2011) aims at sampling from π by augmenting the target distribution with an auxiliary

Algorithm 1: Hamiltonian Monte Carlo

Input: Starting point θ_0, step size ϵ, number of leapfrog steps L, covariance matrix M, number of iterations N.
Output: sample $\{\theta_0, \ldots, \theta_N\}$ drawn from $\pi(\theta)$.

for $n \leftarrow 1$ **to** N **do**
  draw v from $\mathcal{N}(\cdot \mid 0, M)$
  compute $(\theta^\star, v^\star) = \texttt{Leapfrog}(\theta_{n-1}, v, \epsilon, L)$ **compute**

$$\rho = 1 \wedge \exp\left\{H(\theta_{n-1}, v) - H\left(\theta^\star, -v^\star\right)\right\}$$

  set

$$(\theta_n, v_n) = \begin{cases} (\theta^\star, -v^\star), & \text{with probability } \rho \\ (\theta_{n-1}, v), & \text{otherwise.} \end{cases}$$

end
return $(\theta_0, \ldots, \theta_N)$

Function `Leapfrog(`θ, v, ϵ, L`)`
  compute $(\theta_0, v_0) = (\theta, v - \epsilon/2 \nabla U(\theta))$
  for $\ell \leftarrow 1$ **to** L **do**
    compute $\theta_\ell = \theta_{\ell-1} + \epsilon M^{-1} v_{\ell-1}$
    compute $v_\ell = v_{\ell-1} - \epsilon \nabla U(\theta_\ell)$
  end
  return $(\theta_L, v_L + \epsilon/2 \nabla U(\theta_L))$

variable $v \in \mathbb{R}^d$, referred to as momentum variable (as opposed to θ being called the position), distributed according to a d-dimensional Normal distribution, $\mathcal{N}(\cdot \mid 0, M)$. Thus, HMC samples from the joint $\pi(\theta, v) = \pi(\theta)\mathcal{N}(v \mid 0, M)$, which marginal chain in θ is the distribution of interest. We refer the reader to Betancourt (2018) for a discussion on HMC and the choice of other momentum distributions.

We define the Hamiltonian as the unnormalised negative joint log-density

$$H(\theta, v) = U(\theta) + \frac{1}{2} v^T M^{-1} v.$$

HMC generates proposals for θ based on the Hamiltonian dynamics as

$$\frac{\mathrm{d}\theta}{\mathrm{d}t} = \frac{\partial H}{\partial v} = M^{-1} v \quad \text{and} \quad \frac{\mathrm{d}v}{\mathrm{d}t} = -\frac{\partial H}{\partial \theta} = -\nabla U(\theta). \tag{1}$$

Such a mechanism aims at efficiently exploring the parameter space $\Theta \times \mathbb{R}^d$ as compared to standard random-walk Metropolis-Hastings proposals while preserving the measure $\pi(\mathrm{d}\theta, \mathrm{d}v)$ – the solution flow being reversible. Since the dynamic preserves the Hamiltonian, decreases in the potential function U are compensated by increases in its momentum and conversely, which facilitates moves between low and high probability regions. In practice, the above differential equations cannot be solved analytically and HMC samplers resort to time reversible and symplectic numerical integrators, that is integrators which conserve the volume. Among these,

the leapfrog integrator is commonly used for its trade-off between second order accuracy and computational cost. Given a discretisation time-step ϵ, it yields a map $F_\epsilon : (\theta, v) \mapsto (\theta^\star, v^\star)$ defined by

$$\begin{cases} \theta^\star = \theta + \epsilon M^{-1} r, \\ v^\star = r - \epsilon/2 \nabla U(\theta^\star), \end{cases} \qquad \text{where} \quad r = v - \epsilon/2 \nabla U(\theta).$$

It corresponds to a three stage procedure that alternatively updates v and θ to approximate the solution at time ϵ, and to approximate solution at a time t, one repeats the procedure $L = \lfloor t/\epsilon \rfloor$ times. L is referred to as number of leapfrog steps. This discrete scheme does no longer leave the measure $\pi(\mathrm{d}\theta, \mathrm{d}v)$ invariant. To account for the discretisation bias and preserve the targetted measure, an accept-reject step is introduced (see Algorithm 1). A transition from (θ, v) to a proposal $\left(\theta^\star, -v^\star\right)$ corresponding to the approximated solution for an integration time $L\epsilon$ is accepted with probability

$$\rho\left(\theta, v, \theta^\star, v^\star\right) = 1 \wedge \exp\left\{H(\theta, v) - H\left(\theta^\star, -v^\star\right)\right\}. \tag{2}$$

which implies that detailed balance is satisfied for the target $\pi(\mathrm{d}\theta, \mathrm{d}v)$. The solution of (1) keeps the Hamiltonian constant, meaning the proposals are always accepted for the exact dynamic. After discretisation, the deviation to $H(\theta, v)$ can be bounded (Leimkuhler and Reich, 2004),

$$\left|H(\theta, v) - H\left(\theta^\star, -v^\star\right)\right| < C\epsilon^2.$$

Hence, the acceptance rate in Algorithm 1 still tends to be high even for a proposal $\left(\theta^\star, -v^\star\right)$ quite far from (θ, v), provided ϵ remains moderate. In practice, the choice of ϵ can be done on the basis of the global acceptance rate of the sampler using the primal-dual averaging method (Nesterov, 2009). The latter provides an adaptive MCMC scheme which aims at a targeted acceptance probability $p_0 \in (0, 1)$. Under mild conditions, Beskos et al. (2013) have shown that the optimal scaling for HMC yields an acceptance probability of around 0.65. Though it is not necessarily a suitable choice for p_0 on complex model, it can serve as a useful benchmark value.

We now briefly introduce the NUTS algorithm, referring the reader to Hoffman and Gelman (2014) for more details. NUTS is a version of HMC sampler that eliminates the need to specify the number L of integration steps by adaptively choosing the locally largest value at each iteration of the algorithm. More precisely, given a step size ϵ, the current value of θ and a momentum v resampled from a Gaussian distribution $\mathcal{N}(0, M)$, an iteration of NUTS begins with mimicking the Hamiltonian dynamics by recursively doubling the leapfrog path, either forward or backward with equal probability, until the path begins to retrace towards the starting point. For Euclidean version, this means that the backward and forward end points of the path, (θ^-, v^-) and (θ^+, v^+), satisfy

$$(\theta^+ - \theta^-) \cdot M^{-1} v^- < 0 \quad \text{or} \quad (\theta^+ - \theta^-) \cdot M^{-1} v^+ < 0\,. \tag{3}$$

To extend the aforementioned termination criterion to the non-Euclidean case, we refer the reader to Betancourt (2018).

Denote $\mathcal{E}$ the set of the position-momentum pairs visited along the forward and backward leapfrog path. The second step of NUTS consists in randomly picking one of the pairs in $\mathcal{E}$, each pair being weighted according to the target density. In its

original version, the latter was done by a slice sampling move. Given a realisation u from the uniform distribution on $[0, 1]$, one draws uniformly in the following set of points

$$\{(\theta^\star, v^\star) \in \mathcal{E} \mid H(\theta, v) - H(\theta^\star, v^\star) \geq \log u\}.$$

The overall process allows for detailed balance to hold and thus validates the NUTS algorithm. Notice that the current version implemented in Stan (Stan Development Team, 2018b; Betancourt, 2018) uses a multinomial sampling strategy instead of the slice sampling one.

Compared with the standard HMC sampler, which uses a fixed length L across iterations, NUTS relies on an adaptive scheme while requiring an evaluation of the Hamiltonian along its entire leapfrog path. Even though the leapfrog path of NUTS is locally the longest possible in terms of Equation (3), we note that the final position-momentum pair is almost never one of the two endpoints of the generated leapfrog path. Therefore the distance between the position of the proposed value and the current one is smaller than it could be, which induces a waste of computation time.

3 Learning leapfrog scale

In what follows, we explore the opportunity to use a termination criterion similar to Equation (3) for choosing L in the HMC sampler based on the empirical distribution of the longest integration times. In what follows, the step size ϵ is assumed to be chosen using the primal-dual averaging method which is the golden standard also implemented for NUTS.

3.1 Empirical distribution of leapfrog scale

Given a step size ϵ and a starting point (θ, v), denote $(\theta^\star_\ell, v^\star_\ell) = \texttt{Leapfrog}(\theta, v, \epsilon, \ell)$, the value of the pair after ℓ iterations of the leapfrog integrator, $\ell \in \mathbb{N}$. We define the *longest batch* $L_\epsilon(\theta, v)$ associated with (θ, v, ϵ) as the first $\ell \in \mathbb{N}$ such that $(\theta^\star_\ell - \theta) \cdot M^{-1} v^\star_\ell < 0$, referred to as a U-turn condition in what follows. We can construct an empirical marginal distribution of the longest batch by running K iterations of HMC with the optimised step size and an initial number of leapfrog steps L_0 (see Algorithm 2). Specifically, each iteration of HMC is run for L_0 leapfrog steps to generate the next state of the Markov chain. For this iteration, and the current state of the chain, the longest batch can be produced. Namely, if $L_\epsilon(\theta, v) \leq L_0$ this value can be identified and stored; otherwise the integration scheme is performed until the U-turn condition applies. This scheme obviously produces a sequence of longest batches

$$\mathcal{L} = \left\{ L_\epsilon\left(\theta_0, v^{(1)}\right), \cdots, L_\epsilon\left(\theta_{K-1}, v^{(K)}\right) \right\},$$

where θ_{i-1} and $v^{(i)}$, $i = 1, \ldots, K$, denote respectively the current value for the θ-chain and the resampled momentum at iteration i of the HMC sampler. The corresponding empirical probability measure is

$$\hat{P}_{\mathcal{L}} = \frac{1}{K} \sum_{k=0}^{K-1} \delta_{L_\epsilon(\theta_k, v^{(k+1)})},$$

where δ_ℓ denotes the Dirac measure at ℓ.

Algorithm 2: Longest batch empirical distribution

Input: Starting point θ_0, step size ϵ, number of leapfrog steps L_0, covariance matrix M, number of iterations K.

Output: sequence $\mathcal{L} = (\mathcal{L}_1, \ldots, \mathcal{L}_K)$ of longest batches.

for $k \leftarrow 1$ **to** K **do**

  draw $v^{(k)}$ from $\mathcal{N}(\cdot \mid 0, M)$

  `/* Computing` $\mathcal{L}_k = L_\epsilon(\theta_{k-1}, v^{(k)})$ `*/`

  compute

$$(\theta^\star, v^\star, \mathcal{L}_k) = \texttt{LongestBatch}(\theta_{k-1}, v^{(k)}, \epsilon, L_0)$$

  if $\mathcal{L}_k < L_0$ **then**

    compute $(\theta^\star, v^\star) = \texttt{Leapfrog}(\theta^\star, v^\star, \epsilon, L_0 - \mathcal{L}_k)$

  end

  compute

$$\rho = 1 \wedge \exp\left\{H(\theta_{k-1}, v^{(k)}) - H\left(\theta^\star, -v^\star\right)\right\}$$

  set

$$(\theta_k, v_k) = \begin{cases} (\theta^\star, -v^\star), & \text{with probability } \rho \\ (\theta_{k-1}, v^{(k)}), & \text{otherwise.} \end{cases}$$

end

return $(\mathcal{L}_1, \ldots, \mathcal{L}_K)$

Function `LongestBatch`$(\theta,\ v,\ \epsilon,\ L)$

  set $\ell = 0$; $(\theta^+, v^+) = (\theta, v)$; $(\theta^\star, v^\star) = (\theta, v)$

  while $(\theta^+ - \theta) \cdot M^{-1} v^+ \geq 0$ **do**

    set $\ell = \ell + 1$

    compute $(\theta^+, v^+) = \texttt{Leapfrog}(\theta^+, v^+, \epsilon, 1)$

    if $\ell == L$ **then**

      set $(\theta^\star, v^\star) = (\theta^+, v^+)$

    end

  end

  return $(\theta^\star, v^\star, \ell)$

3.2 Convergence Analysis

Let $Q_\epsilon(\cdot \mid \theta, v)$ be the distribution of the longest batch conditionally on the current state (θ, v) with step size ϵ and Q_ϵ be its marginal distribution with respect to the augmented target $\pi(\mathrm{d}\theta, \mathrm{d}v)$. The transition kernel of the HMC sampler can be written as the composition of two Markov kernels P_1 and P_2 over $\Theta \times \mathbb{R}^d$:

$$\begin{cases} P_1((\theta, v), (\mathrm{d}\theta, \mathrm{d}v)) = \delta_\theta(\mathrm{d}\theta)\mathcal{N}(\mathrm{d}v \mid 0, M) \\ \\ P_2((\theta, v), (\mathrm{d}\theta, \mathrm{d}v)) \\ \qquad = \rho\delta_{(\theta^\star, -v^\star)}(\mathrm{d}\theta, \mathrm{d}v) + (1-\rho)\delta_{(\theta, v)}(\mathrm{d}\theta, \mathrm{d}v), \end{cases}$$

where $(\theta^\star, v^\star) = \texttt{Leapfrog}(\theta, v, \epsilon, L)$ and ρ is defined by Equation (2). Both kernels are reversible with respect to the augmented target $\pi(\mathrm{d}\theta, \mathrm{d}v)$ and hence the composi-

Algorithm 3: Empirical Hamiltonian Monte Carlo (eHMC)

Input: Starting point θ_0, step size ϵ, empirical dis-tribution of longest batches $\hat{P}_{\mathcal{L}}$ (generated by Algorithm 2), covariance matrix M, number of iterations N.
Output: sample $\{\theta_0, \ldots, \theta_N\}$ drawn from $\pi(\theta)$.

for $n \leftarrow 1$ **to** N **do**
  draw v from $\mathcal{N}(\cdot \mid 0, M)$
  draw L from $\hat{P}_{\mathcal{L}}$
  compute $(\theta^\star, v^\star) = \texttt{Leapfrog}(\theta_{n-1}, v, \epsilon, L)$ **compute**

$$\rho = 1 \wedge \exp\left\{H(\theta_{n-1}, v) - H\left(\theta^\star, -v^\star\right)\right\}$$

  set

$$(\theta_n, v_n) = \begin{cases} (\theta^\star, -v^\star), & \text{with probability } \rho \\ (\theta_{n-1}, v), & \text{otherwise.} \end{cases}$$

end
return $(\theta_0, \ldots, \theta_N)$

tion $P_2 \circ P_1$ leaves the latter invariant. As a consequence, $\hat{P}_{\mathcal{L}}$ produced by Algorithm 2 converges to Q_ϵ.

3.3 Empirical Hamiltonian Monte Carlo

We introduce a version of HMC, called empirical HMC (eHMC) that makes use of the empirical distribution $\hat{P}_{\mathcal{L}}$ constructed in Algorithm 2.

At each iteration NUTS derives a locally adapted scale for L that it employs in a probabilistically valid manner, that is, in order to preserve the stationary distribution. Indeed, using directly the longest batch produced at each iteration, albeit interesting in potentially increasing the jumping distance between the current and the proposed states, creates a bias in the resulting stationary distribution. As mentioned earlier, NUTS requires a balanced binary tree in order to preserve the reversibility of the resulting Markov chain. Here we set an orthogonal improvement called eHMC by considering instead a global distribution on the batch size that is consistent with the target distribution.

The core idea of eHMC is to randomly pick a number of leapfrog steps according to the empirical distribution $\hat{P}_{\mathcal{L}}$ at each iteration (see Algorithm 3). The resulting algorithm is valid since the choice of L is independent from the current state of the chain. Thus the resulting transition kernel can be seen as a composition of multiple Markov kernels attached to the same stationary distribution $\pi(\mathrm{d}\theta, \mathrm{d}v)$ (Tierney, 1994).

3.4 Partially refreshed Hamiltonian Monte Carlo

As a follow-up, we mention an accelerated version of HMC, referred to as prHMC, obtained by only refreshing the auxiliary momentum v at some selected iterations. In

order to simplify the description of the algorithm, we introduce the following three Markov transition kernels. The first one corresponds to the Metropolis adjusted Hamiltonian kernel

$$P_1^{\epsilon,L}\left((\theta,v),(d\theta',dv')\right) = \rho\delta_{\{(\theta^\star,-v^\star)\}}(d\theta',dv') + (1-\rho)\delta_{\{(\theta,v)\}}(d\theta',dv'),$$

where $(\theta^\star, v^\star) = \texttt{Leapfrog}(\theta, v, \epsilon, L)$ and ρ is the acceptance probability (2). We then define a resampling kernel

$$P_2\left((\theta,v),(d\theta',dv')\right) = \delta_{\{\theta\}}(d\theta')\mathcal{N}(dv' \mid 0, M)$$

and a flipping kernel

$$P_3\left((\theta,v),(d\theta',dv')\right) = \delta_{\{\theta\}}(d\theta')\delta_{\{-v\}}(dv').$$

It is easy to verify that all three kernels are reversible with respect to $\pi \otimes \varphi$ and hence that $P_3 \circ P_1^{\epsilon,L}$ and $P_1^{\epsilon,L} \circ P_2$ admit $\pi \otimes \varphi$ as their stationary distribution. Furthermore, since $\pi \otimes \varphi$ is symmetric with respect to v, we can obtain that $\pi \otimes \varphi$ is a stationary distribution of $P_3 \circ P_1^{\epsilon,L} \circ P_2$. As a result, the mixture kernel

$$P^{\epsilon,L,\eta} = (1-\eta)P_3 \circ P_1^{\epsilon,L} + \eta P_3 \circ P_1^{\epsilon,L} \circ P_2$$

is a Markov kernel with stationary distribution $\pi \otimes \varphi$ for any $\eta \in [0,1]$. Compared with standard HMC, this alternative Markov kernel resamples the momentum with probability η at each iteration. Besides, $P^{\epsilon,L,\eta}$ conserves the momentum if the proposal is accepted and flips it otherwise. Intuitively, this algorithm potentially gains efficiency over standard HMC for two reasons: leveraging intermediate proposals along leapfrog path when we conserve the momentum and recycling the obtained proposals when the momentum flips (see Algorithm 4 in Appendix). We shall remark that the latter algorithm will eventually explore the level set both forward (that is in the direction of the resampled momentum referred to as $\sigma = 1$) and backward ($\sigma = -1$). For computational purpose, the pairs visited are stored in a matrix w^+. We use the convention that all the momenta visited are stored according to the forward direction requiring to flip the sign of the momentum visited in the backward direction. As a consequence, when proposing a pair stored in w^+, the acceptance probability (2) needs also to be written with respect to the direction σ to ensure the momentum points in the right direction.

4 Numerical Experiment

Throughout the following section, we produce empirical comparisons of the performance of NUTS, eHMC and prHMC samplers in terms of the effective sample size, the expected squared jumped distance or the Kolmogorov-Smirnov distance over several models with increasing complexity. We use the version of NUTS implemented in `Rstan` (Stan Development Team, 2018a). The code for eHMC and prHMC samplers as well as the data generated in what follows are available online at https://github.com/jstoehr/eHMC. In order to differentiate between the computing

requirements of the compared samplers, it seems fairer to evaluate samples generated under the same computation budget. As a result, the ESS and the ESJD are calibrated by the evaluation cost of the gradient of the logarithm of target density function. This choice is based on the assumption that the largest portion of the computation budget is dedicated to compute ∇U.

The effective sample size (ESS). We recall that the standard effective sample size associated with a Markov chain $\{\theta_1, \ldots, \theta_N\}$ is defined as

$$\mathrm{ESS} = N \Big/ \left(1 + 2\sum_{k>0} \rho_k\right),$$

where ρ_k is the lag k auto-correlation of the Markov chain. Informally, the ESS of an MCMC output is interpretable as providing the equivalent number of independent simulations from the target distribution, where equivalent means with equivalent variance. The following examples resort to the `ess` function of the R package `mcmcse` (Flegal et al., 2016) to produce estimates of the ESS for each component of the parameter. Hence, the more efficient a sampler is in its approximation to *i.i.d.* sampling from the target, the larger the associated ESS should be. The evaluations below report the *minimum* of ESS per gradient evaluation when the minimum is taken over all components of the parameter of the model, a measure that identifies the least effective direction for the sampler.

The expected squared jumped distance (ESJD). Furthermore, considering that the ESS criterion only reflects on the marginal efficiency of a sampler, we report in addition the now traditional expected squared jump distance. While depending on the measure of the jump, this quantity appears as a global mixing assessment that expresses another aspect of the efficiency of a sampler. We recall that, given a set of generated samples $\{\theta_1, \ldots, \theta_N\}$, its estimated ESJD is defined as

$$\mathrm{EJSD} = \frac{1}{N-1} \sum_{n=1}^{N-1} \|\theta_{n+1} - \theta_n\|_2^2 .$$

The Kolmogorov-Smirnov distance (KS). We also compare the accuracy between NUTS, eHMC, and prHMC using the Kolmogorov-Smirnov (KS) distance. For two one-dimensional distributions with cumulative distribution functions F and G, their KS distance is defined as

$$\mathrm{KS}(F, G) = \sup_{x \in \mathbb{R}} |F(x) - G(x)| .$$

The choice of a reference distribution, namely the target distribution, is obviously paramount to measure the performance of different samplers. We therefore generate a large sample by NUTS ($50,000$ sample points with targeted probability $p_0 = 0.95$) and use its empirical distribution as a surrogate for the target to compute the KS distances in each example. The KS distance being solely available component-wise, we report the maximal KS distance across the parameters components, which reflects the most constrained direction of each sampler.

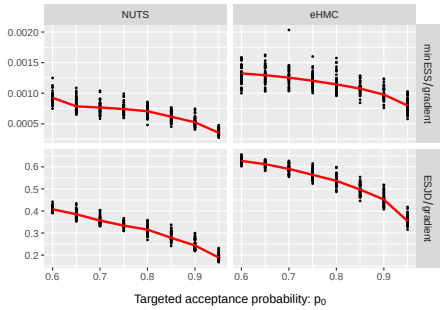


Fig. 1 Comparison between NUTS and eHMC samplers for a multivariate Normal distribution target: the x-axis is indexed by the targeted acceptance probability, p_0, associated with the primal-dual averaging step, and the y-axis in both plots corresponds to the minimum ESS across all parameter coordinates (first row) and to the ESJD (second row) per evaluation of ∇U, respectively. A black dot indicates the performance for a single experiment and the red curve is the resulting median of the 40 black-dot values for a particular p_0.

In each of the following models, the matrix M is set to an adaptively tuned diagonal mass matrix in the Stan NUTS runs. This diagonal matrix is then used in the eHMC runs. Given a targeted acceptance probability p_0 which evenly ranges from 0.6 to 0.95, both NUTS and eHMC results are based on $N = 25,000$ iterations, with the first $5,000$ iterations used to tune the corresponding step size ϵ. Once ϵ is tuned, Algorithm 2 is run over 2,000 iterations to construct the set $\mathcal{L}$ which is then exploited by eHMC. When considered, the KS distance is solely computed for the step size ϵ corresponding to $p_0 = 0.8$. The various results were obtained over 40 repetitions of each experiment.

4.1 Multivariate Normal Distribution

The first example is a toy example that aims at a multivariate Normal distribution with zero mean and covariance matrix A as its target,

$$\pi(\theta) \propto \exp\left\{-\theta^T A^{-1}\theta/2\right\}$$

The dimension of the vector is chosen to be 100, with

$$A_{i,j} = 0.99^{|i-j|}, \quad i,j = 1,\cdots,100$$

Figure 1 displays the minimum ESS across the parameter components and the ESJD per gradient for each sampler across the range of p_0 based on the 40 repetitions. As is clear from the plots, the gain brought by using eHMC is important on most of the range of p_0's, without ever exhibiting a significant loss of efficiency, relative to NUTS. In addition, the boxplots on Figure 2 demonstrate the further improvement brought by prHMC over eHMC since the maximum KS distance is uniformly smaller across experiments.

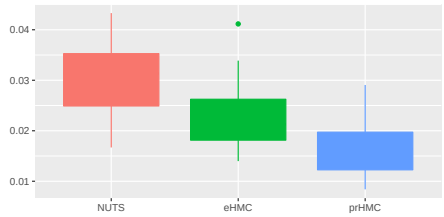


Fig. 2 Comparison of NUTS, eHMC and prHMC over multivariate Normal model in terms of Kolmogorov-Smirnov distance. The y-axis scales the maximum KS distance over all components based on 40 replications and $p_0 = 0.8$.

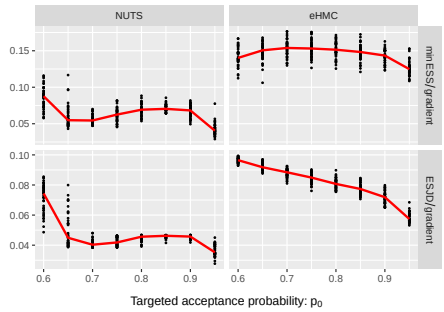


Fig. 3 Comparison between NUTS and eHMC samplers for a Bayesian logistic model applied to German credit data, with the same conventions as in Fig. 1.

4.2 Bayesian Logistic Posterior

This example target is the posterior distribution of a Bayesian logistic regression under a flat prior $\pi_0(\theta) \propto 1$,

$$\pi(\theta) \propto \prod_{n=1}^{N} \frac{\exp(y_i x_i^T \theta)}{1 + \exp(x_i^T \theta)}$$

The inference is based on a German credit dataset obtained from UCI repository (Frank and Asuncion, 2010), which contains $1,000$ observations and 24 covariates. As described by Hoffman and Gelman (2014), all covariates are normalized to enjoy zero mean and unit variance. Figure 3 displays the minimum ESS across all dimensions and the ESJD per gradient when p_0 varies. In this high dimensional case, the improvement brought by eHMC is stronger, with an ESS up to two-fold for all values of p_0. Once again the boxplots of the KS distance in Figure 4 show the successive improvements of both eHMC and prHMC.

4.3 Stochastic Volatility Model

This example aims at analysing a posterior distribution associated with a stochastic volatility model and with T artificial (generated) observations. The stochastic

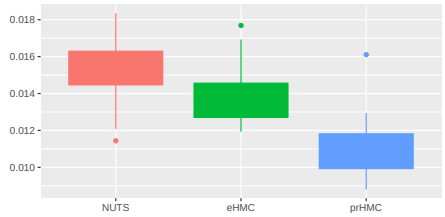


Fig. 4 Comparison of NUTS, eHMC and prHMC applied to German credit data in terms of Kolmogorov-Smirnov distance. The y-axis scales the maximum KS distance over all components based on 40 replications and $p_0 = 0.8$.

volatility model is based on the state-space equations:

$$x_1 \sim \mathcal{N}\left(0, \frac{\sigma^2}{1-\varphi^2}\right),$$

for $t = 1, 2, \cdots, T$,

$$\begin{cases} y_t &= \epsilon_t \kappa \exp(0.5 x_t), \\ \epsilon_t &\sim \mathcal{N}(0,1), \end{cases}$$

and for $t = 2, \cdots, T$

$$\begin{cases} x_t &= \varphi x_{t-1} + \eta_t, \\ \eta_t &\sim \mathcal{N}(0, \sigma^2). \end{cases}$$

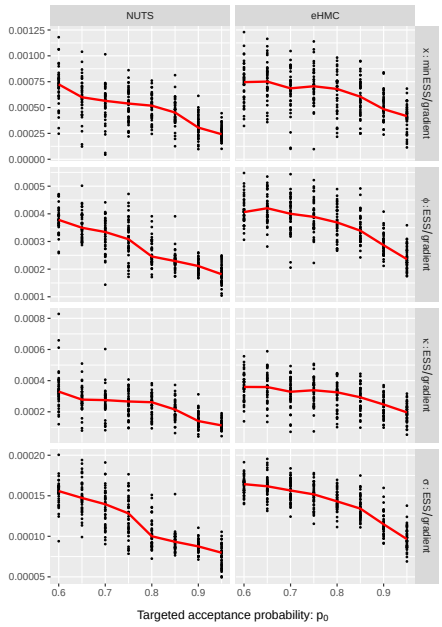
where only the y_t's are observed. The prior is chosen to be $p(\kappa) \propto 1/\kappa$, $0.5(1+\varphi) \sim \text{Beta}(20, 1.5)$, and $\sigma^{-2} \sim \chi^2(10, 0.05)$. In order to reduce overflow problems, we reparametrize the three parameters φ, κ and σ^2 as

$$\alpha = \log\left(\frac{1+\varphi}{1-\varphi}\right), \quad \beta = \log(\kappa) \quad \text{and} \quad \gamma = \log(\sigma^2).$$

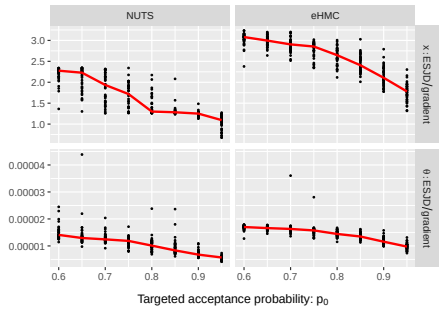
Following Girolami and Calderhead (2011), we generate $T = 1,000$ observations with parameters chosen as $\varphi_0 = 0.98$, $\kappa_0 = 0.65$ and $\sigma_0 = 0.15$. The dimension of the simulation space is thus $T + 3 = 1,003$, with $1,000$ latent variables $\{x_t\}_{t=1}^T$. The associated potential function is given by

$$\begin{aligned} &U(\alpha, \beta, \gamma, \mathbf{x}|\mathbf{y}) \\ &= T\beta + \sum_{t=1}^{T} \frac{x_t}{2} + \sum_{t=1}^{T} \frac{y_t^2}{2\exp(2\beta)\exp(x_t)} - 20.5\alpha \\ &\quad + 22.5 \log(e^\alpha + 1) + \left(\frac{T}{2} + 5\right)\gamma + \frac{2x_1^2 e^\alpha}{(e^\alpha+1)^2 e\gamma} \\ &\quad + \frac{1}{2}\sum_{t=2}^{T}\left\{e^{-\gamma}(x_t - \frac{e^\alpha - 1}{e^\alpha + 1}x_{t-1})^2\right\} + \frac{1}{4e^\gamma}. \end{aligned}$$

Figure 5(a) compares the minimum ESS per evaluation of ∇U of each parameter group and Figure 5(b) shows the ESJD on $\{x_t\}_{t=1}^T$ and $\theta = (\phi, \kappa, \sigma)$ per evaluation of ∇U for each sampler over the range of targeted acceptance probability p_0. In this high dimensional model, eHMC (and its variants not described here) perform at least as well as NUTS over each experiment configuration.



(a)



(b)

Fig. 5 Comparison between NUTS and eHMC for the stochastic volatility model. The x-axis is the acceptance probability p_0 fed into primal-dual averaging. A black dot displays the performance for a single experiment and the red curve is the resulting median over the 40 y-values and for a particular p_0. (a) The y-axis corresponds to the minimum ESS across components of $\{x_t\}_{t=1}^T$ and to the ESS for each parameter component per evaluation of ∇U, respectively. (b) The y-axis corresponds to the ESJDs of $\{x_t\}_{t=1}^T$ and of $\theta = (\phi, \kappa, \sigma)$ per evaluation of ∇U, respectively.

4.4 Logistic Item Response Theory Model

This example considers a hierarchical item response theory (IRT) model (Hambleton et al., 1991), written as, for $i = 1, 2, \ldots, I$ and $j = 1, 2, \cdots, J$,

$$\text{logit}(\mathbb{P}(y_{i,j} = 1 | \eta, a, b)) = a_i(\eta_j - b_i)$$

Table 1 Comparison between NUTS and eHMC for a two parameter logistic item response theory model: each entry corresponds to an average over 40 experiments of the maximum value across the possible values of p_0 of the minimum ESS per evaluation of ∇U over each group of parameters.

Sampler	ESS ($\times 10^{-2}$) per evaluation (mean $\pm$ sd)		
	$\{\eta_j\}_{j=1}^{100}$	$\{a_i\}_{i=1}^{20}$	$\{b_i\}_{i=1}^{20}$
NUTS	1.51 ± 0.19	1.13 ± 0.15	1.20 ± 0.14
eHMC	2.22 ± 0.23	1.63 ± 0.13	1.83 ± 0.15

with

$$\eta_j \sim \mathcal{N}(0, \sigma_\eta^2) \quad \text{and} \quad \begin{cases} a_i \sim \mathcal{LN}(0, \sigma_a^2), \\ b_i \sim \mathcal{LN}(\mu_b, \sigma_b^2). \end{cases}$$

where $y_{i,j}$ is the response for person j to item i, a_i and b_i are referred to as the discrimination and difficulty parameters, respectively, and η_j is called the ability parameter. The priors for the hyperparameters are $\sigma_\eta \sim \text{Cauchy}(0, 2)$, $\sigma_a \sim \text{Cauchy}(0, 2)$, $\mu_b \sim \mathcal{N}(0, 25)$, and $\sigma_b \sim \text{Cauchy}(0, 2)$ under the constraints $\sigma_\eta > 0$, $\sigma_a > 0$, $\sigma_b > 0$ and $a_i > 0$. These constraints are handled in HMC by looking at the logarithm transforms with the appropriate Jacobian adjustment. We use a benchmark dataset from STAN repository (http://tinyurl.com/y4w2zlyj), in which $I = 20, J = 100$.

As shown in Table 1, the maximum of the minimum ESS over the 40 experiment is on average around 1.5 times larger for eHMC, depending on the group of interest. Note that this evaluation of efficiency is based on the minimum across components of the parameter subset and on the maximum across different values of p_0 for each of the methods examined and for each of the repetitions of the Monte Carlo experiment. Mean and variance in this table are thus taken over 40 values corresponding to possibly different values of p_0.

4.5 Susceptible-Infected-Recovered Model

Susceptible-infected-recovered (SIR) models are used to represent disease transmission, for epidemics like cholera, within a population. The model in this example is, for $k = 1, \cdots, N_t$,

$$\begin{cases} \eta_k \sim \text{Poisson}(y_{t_{k-1},1} - y_{t_k,1}), \\ \hat{B}_k \sim \mathcal{LN}(y_{t_k,4}, 0.15^2). \end{cases}$$

where the dynamic of $y_t \in \mathbb{R}^4$ is given by

$$\frac{dy_{t,1}}{dt} = -\frac{\beta y_{t,4}}{y_{t,4} + \kappa_0} y_{t,1}, \quad \frac{dy_{t,2}}{dt} = \frac{\beta y_{t,4}}{y_{t,4} + \kappa_0} y_{t,1} - \gamma y_{t,2},$$
$$\frac{dy_{t,3}}{dt} = \gamma y_{t,3}, \quad \frac{dy_{t,4}}{dt} = \xi y_{t,2} - \phi y_{t,4}.$$

Table 2 Comparison between NUTS and eHMC for a SIR model: same legend as for Table 1.

Sampler	ESS ($\times 10^{-3}$) per evaluation (mean $\pm$ sd)			
	β	γ	ξ	ϕ
NUTS	6.26 $\pm$0.78	6.25 $\pm$0.76	7.71 $\pm$0.66	7.38 $\pm$0.88
eHMC	16.2 $\pm$1.8	15.8 $\pm$1.5	20.7 $\pm$2.1	20.6 $\pm$1.9

where $y_{t,1}, y_{t,2}, y_{t,3}$ represent, respectively, the number of susceptible, infected, and recovered people within the community, and $y_{t,4}$ represents the concentration of the virus in the water reservoir used by this population (Grad et al., 2012). In this model, the size of the population $y_{t,1} + y_{t,2} + y_{t,3}$ is assumed to be constant, which means that there is neither birth nor death resulting from the disease or from other reasons. The parameter η_k represents the size of the population that becomes infected from being susceptible during the time interval $[t_{k-1}, t_k]$.

We used the dataset available on STAN repository (http://tinyurl.com/y69t3sqs). The latter contains $N_t = 20$ observations $\{\eta_k, \hat{B}_k\}_{k=1}^{N_t}$ over the time steps $t_k = 7k$. We set the priors over the parameters as follows:

$$\beta \sim \mathcal{C}(0, 2.5), \quad \xi \sim \mathcal{C}(0, 25),$$
$$\gamma \sim \mathcal{C}(0, 1), \quad \phi \sim \mathcal{C}(0, 1);$$

In order to accomodate the constraints $\beta > 0$, $\gamma > 0$, $\xi > 0$ and $\phi > 0$, we used a reparametrisation into logarithms. Table 2 reports the best performance of each sampler across the possible values of p_0. These experiments show once again that eHMC exhibit a higher efficiency than NUTS in terms of ESS. Besides, Figure 6 exhibits a reduction in the maximum KS distance when using eHMC over NUTS and prHMC over eHMC for the same computation cost. Table 2 suggests less simulations are actually needed to achieve the same precision by eHMC.

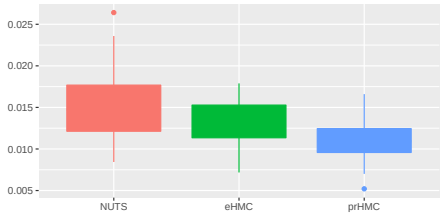


Fig. 6 Comparison of NUTS, eHMC and prHMC over SIR model in terms of Kolmogorov-Smirnov distance. The y-axis scales the maximum KS distance over all components based on 40 replications and $p_0 = 0.8$.

5 Conclusion and Prospective

Our eHMC algorithm is an HMC algorithm elaborating on NUTS that can been seen as an alternative, in that it relies on an empirical version, $\hat{P}_{\mathcal{L}}$, of the distribution of the longest batch size generated in the leapfrog integrator, for a given step size ϵ. Our motivation for this version is to reduce wasted simulations created by NUTS, waste that results from a stationarity constraint and the induced backtracking of NUTS. Even though the length and cost of the path is *de facto* reduced for eHMC, we have shown through several Monte Carlo experiments that this version leads to some degree of improved efficiency over the standard NUTS, when evaluated by either ESS or ESJD per potential gradient evaluation. While we do not aim here at optimising the length L of the HMC step in terms of one of these criteria, the automated random choice of L at each iteration shows clear promises.

The comparisons produced here may appear to contradict those of Hoffman and Gelman (2014). The reason for this opposition is that Hoffman and Gelman (2014) compared NUTS with the standard HMC sampler, which number of steps L is fixed at each and every iteration, instead of an HMC with a random number of steps. It has already been observed in the literature that resorting to a random integration time in an HMC sampler should improve performances.

The improvement of eHMC over NUTS proceeds from two features. First, eHMC chooses a set of globally appropriate values of L for an HMC sampler. Second, as noted above, NUTS wastes computation effort to some extent. In fact, even though NUTS only generates batches towards preventing from retracing at each iteration, it is impossible to ensure that the current position is close to an end-point of the generated leapfrog path. Furthermore, the NUTS proposal, although not uniformly picked up from the candidate set, is rarely one of the end-points in this set. Both limitations restrict the distance between the current position and the proposal.

Obviously, $\hat{P}_{\mathcal{L}}$ contains valuable information about the efficient number of steps in the leapfrog integrator for HMC samplers and the related distribution deserves further investigation. Besides, we can also insert Algorithm 2 into a primal-dual averaging scheme to learn a rough version of $\mathcal{L}$ and hence save further computation cost.

Further directions along that theme worth exploring are designing a distribution for the integration time that depends on the Hamiltonian level and integrating a Rao-Blackwellisation step as in Nishimura and Dunson (2015).

Acknowledgements The authors are grateful to Bob Carpenter and Matt Graham for comments on this earlier version.

References

Beskos A, Pillai N, Roberts G, Sanz-Serna JM, Stuart A (2013) Optimal tuning of the hybrid Monte Carlo algorithm. Bernoulli 19(5A):1501–1534

Betancourt M (2018) A conceptual introduction to Hamiltonian Monte Carlo. arXiv preprint arXiv:170102434v2

Betancourt M, Byrne S, Girolami M (2014) Optimizing the integrator step size for Hamiltonian Monte Carlo. arXiv preprint arXiv:14116669

Duane S, Kennedy AD, Pendleton BJ, Roweth D (1987) Hybrid Monte Carlo. Physics Letters B 195(2):216–222

Flegal JM, Hughes J, Vats D (2016) mcmcse: Monte Carlo Standard Errors for MCMC. Riverside, CA and Minneapolis, MN, R package version 1.2-1

Frank A, Asuncion A (2010) UCI Machine Learning Repository. Irvine, CA: University of California. http://archive ics uci edu/ml, School of Information and Computer Science 213:2–2

Girolami M, 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

Grad YH, Miller JC, Lipsitch M (2012) Cholera modeling: challenges to quantitative analysis and predicting the impact of interventions. Epidemiology (Cambridge, Mass) 23(4):523

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

Hambleton RK, Swaminathan H, Rogers HJ (1991) Fundamentals of Item Response Theory. Sage Press, Newbury Park, CA

Hoffman MD, 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

Leimkuhler B, Reich S (2004) Simulating Hamiltonian Dynamics, vol 14. Cambridge University Press

Neal RM (2011) MCMC using Hamiltonian dynamics. Handbook of Markov Chain Monte Carlo 2(11):2

Nesterov Y (2009) Primal-dual subgradient methods for convex problems. Mathematical Programming 120(1):221–259

Nishimura A, Dunson D (2015) Recycling intermediate steps to improve Hamiltonian Monte Carlo. arXiv e-prints arXiv:1511.06925, 1511.06925

Stan Development Team (2018a) RStan: the R interface to Stan. R package version 2.17.3. Http://mc-stan.org

Stan Development Team (2018b) Stan Modeling Language Users Guide and Reference Manual. Version 2.18.0. http://mc-stan.org

Tierney L (1994) Markov Chains for Exploring Posterior Distributions (with Discussion). The Annals of Statistics 22:1701–1762

Wang Z, Mohamed S, De Freitas N (2013) Adaptive Hamiltonian and Riemann Manifold Monte Carlo Samplers. In: Proceedings of the 30th International Conference on Machine Learning, JMLR.org, ICML, vol 28, pp III–1462–III–1470

Appendix

Algorithm 4: Partially refreshed Hamiltonian Monte Carlo (prHMC)

Input: Starting point θ_0, step size ϵ, $\hat{P}_{\mathcal{L}}$ generated by Algorithm 2, covariance matrix M, refreshment probability η, number of iterations N
Output: sample $\{\theta_0, \ldots, \theta_N\}$ drawn from $\pi(\theta)$.

draw v_0 from $\mathcal{N}(0, M)$
set $\omega^+ = (\theta_0, v_0)$; $\sigma = 1$; $i = 1$; $\ell = 1$
for $n \leftarrow 1$ **to** N **do**
  draw u from $\mathcal{U}[0,1]$ and L from $\hat{P}_{\mathcal{L}}$, set $L = \lceil L/3 \rceil$
  if $u < \eta$ **then**
    draw v from $\mathcal{N}(0, M)$
    compute $\omega^+ = \big((\theta_{n-1}, v)^T, \texttt{LFpath}(\theta_{n-1}, v, \epsilon, L)\big)$
    set $\ell = L + 1$ and

$$(\theta_n, v_n, i, \sigma) = \begin{cases} (\omega^+_{1,L+1}, \omega^+_{2,L+1}, \ell, 1), & \text{with probability } \rho(\theta_n, v_n, \omega^+_{1,L+1}, -\omega^+_{2,L+1}) \\ (\theta_{n-1}, -v, 1, -1), & \text{otherwise.} \end{cases}$$

  else
    set $j = i + \sigma L$
    compute $\Delta = (j - \ell)\mathbb{1}_{\sigma=1} - (j-1)\mathbb{1}_{\sigma=-1}$
    if $\Delta > 0$ **then**
      set $\ell = \ell + \Delta$
      compute $\{(\theta^\star_k, v^\star_k)\}_{k=1}^{\Delta} = \texttt{LFpath}(\theta_{n-1}, v_{n-1}, \epsilon, \Delta)$
      if $\sigma = 1$ **then**
        set $\omega^+ = \big(\omega^+, (\theta^\star_1, v^\star_1)^T, \ldots, (\theta^\star_\Delta, v^\star_\Delta)^T\big)$
      else
        set $\omega^+ = \big((\theta^\star_\Delta, -v^\star_\Delta)^T, \ldots, (\theta^\star_1, -v^\star_1)^T, \omega^+\big)$
        set $i = i + \Delta$; $j = 1$
      end
    end
    set

$$(\theta_n, v_n, i, \sigma) = \begin{cases} (\omega^+_{1,j}, \sigma\omega^+_{2,j}, j, \sigma), & \text{with probability } \rho(\theta_{n-1}, v_{n-1}, \omega^+_{1,j}, -\sigma\omega^+_{2,j}) \\ (\theta_{n-1}, -v_{n-1}, i, -\sigma), & \text{otherwise.} \end{cases}$$

  end
end
return $(\theta_0, \ldots, \theta_N)$

Function `LFpath`(θ, v, ϵ, L)
  compute $(\theta_0, v^\star) = (\theta, v - \epsilon/2 \nabla U(\theta))$
  for $\ell \leftarrow 1$ **to** L **do**
    compute $\theta_\ell = \theta_{\ell-1} + \epsilon M^{-1} v^\star$
    compute $v_\ell = v^\star - \epsilon/2 \nabla U(\theta_\ell)$
    compute $v^\star = v_\ell - \epsilon/2 \nabla U(\theta_\ell)$
  end
  return $((\theta_1, v_1), \ldots, (\theta_L, v_L))$