

Hamiltonian Monte Carlo for (Physics) Dummies

Arghya Mukherjee¹ and Dootika Vats¹

¹*Department of Mathematics and Statistics
Indian Institute of Technology Kanpur*

Received: 04 January 2026; Revised: 25 May 2026; Accepted: 27 May 2026

Abstract

Sampling-based inference has seen a surge of interest in recent years. Hamiltonian Monte Carlo (HMC) has emerged as a powerful algorithm that leverages concepts from Hamiltonian dynamics to efficiently explore complex target distributions. Variants of HMC are available in popular software packages, enabling off-the-shelf implementations that have greatly benefited the statistics and machine learning communities. At the same time, the availability of such black-box implementations has made it challenging for users to understand the inner workings of HMC, especially when they are unfamiliar with the underlying physical principles. We provide a pedagogical overview of HMC that aims to bridge the gap between its theoretical foundations and practical applicability. This review article seeks to make HMC more accessible to applied researchers by highlighting its advantages, limitations, and role in enabling scalable and exact Bayesian inference for complex models.

Key words: Gradient-based sampling; Hamiltonian dynamics; No U-turn sampler; Machine learning; Bayesian computation.

AMS Subject Classifications: 65C05, 62F15, 65C40, 65L06

1. Introduction

Markov chain Monte Carlo (MCMC) methods lie at the heart of Bayesian computation, providing a general mechanism to sample from complex posterior distributions that are analytically intractable. Yet, standard algorithms such as the Metropolis–Hastings (Hastings, 1970; Metropolis *et al.*, 1953) or the Gibbs sampler (Geman and Geman, 1984) often struggle in high dimensions or when model parameters exhibit strong correlations, leading to slow mixing and poor scalability in modern applications. Hamiltonian Monte Carlo (HMC), introduced by Duane *et al.* (1987) and made popular by Neal (2011), has been a breakthrough algorithm providing a sampling solution that overcomes many of these hurdles. A standard HMC algorithm introduces auxiliary variables and employs Hamiltonian dynamics to obtain a next candidate draw. Despite its elegance, the practical use of HMC entails many challenges. Effective deployment requires careful tuning of several hyperparameters that

govern both numerical stability and sampling efficiency. Recent investment in off-the-shelf adapted implementations of HMC, like Stan (Carpenter *et al.*, 2017) and PyMC3 (Salvatier *et al.*, 2016), have made it significantly easier for practitioners to employ adaptive HMC algorithms. On the one hand, this has led to renewed interest in Bayesian modeling strategies across disciplines, while on the other hand, HMC and its variants are often treated as black-box algorithms, limiting the opportunities for practitioners to understand and utilize them effectively.

Recently, a few efforts have been made to exposit the details of an HMC algorithm. Neal (2011) presents a great first introduction, Betancourt (2017) provides elegant explanations providing detailed heuristics and pedagogical motivations. Granados and Bañales (2025) focus on explaining HMC using physics fundamentals, and Thomas and Tu (2021) walk a reader through critical implementation details in R. However, arguably, all works assume a certain amount of knowledge of physics. Despite these excellent works, the authors of this paper still found it challenging to intuitively understand the fine interplay between Hamiltonian dynamics and Monte Carlo sampling. Further, it remained unclear how HMC fits into a traditional Metropolis-Hastings setup. This work presents a pedagogical explanation of HMC, introducing Hamiltonian dynamics in a way that, hopefully, does not require much beyond elementary physics. The intended audience is researchers in statistics and machine learning who are interested in sampling methods, understand the fundamentals of Metropolis-Hastings, but, like the authors of this paper, are “physics dummies”.

The rest of the paper is organized as follows. In Section 2, we present the Metropolis-Hastings algorithm and its Jacobian-based extension, the Metropolis-Hastings-Green method, from which we lay the foundation of HMC in Section 3. An effective HMC implementation requires careful choice of tuning parameters, heuristics and theory for which are outlined in Section 4. Section 5 demonstrates the implementation of HMC for a Bayesian model applied to a real dataset. Our approach in these sections is to keep the discussion fairly pedagogical to ensure clarity. In Section 6 we assemble and present recent developments in HMC and its variants, and we end with a discussion in Section 7.

2. Markov chain Monte Carlo

A typical MCMC problem may be posed in the following way. Interest is in obtaining samples from a distribution with density $\pi(x)$, defined on a state space $\mathcal{X} \subseteq \mathbb{R}^d$. (We employ the common abuse of notation, referring to π as both the density and the distribution.) In many Bayesian applications, π represents the posterior density of the model parameters given the observed data. However, the functional form of π often does not correspond to any known distribution, and it is typically available only up to a normalizing constant. That is,

$$\pi(x) \propto \tilde{\pi}(x),$$

such that $\tilde{\pi}(x)$ is completely known and the normalizing constant is possibly unknown. Consequently, obtaining independent samples from π can be difficult even for moderately large d . MCMC provides a general framework for sampling from such intractable target densities by constructing a Markov chain whose stationary and limiting distribution is π . An appropriately constructed MCMC algorithm would ideally be *ergodic*, in that it is guaranteed to converge to π in total variation distance, irrespective of the starting value of the Markov

chain. Although the construction of such algorithms might seem daunting, due to the path-breaking work of Metropolis *et al.* (1953), this is not so challenging. The challenge is to construct algorithms that exhibit this convergence rapidly.

2.1. Metropolis-Hastings algorithm

The Metropolis algorithm originated from a physics application in the 1950s (Metropolis *et al.*, 1953), and was further extended nearly two decades later by Hastings (1970), thus giving rise to the Metropolis–Hastings (MH) algorithm¹. The MH algorithm constructs a Markov chain using a user-chosen proposal distribution that generates a candidate sample based on the current state. This candidate is then accepted or rejected based on a certain probability that guarantees π as the stationary distribution. The specifics are provided in Algorithm 1, which employs a proposal distribution Q with conditional density $q(\cdot | x)$ to generate a candidate sample x^* . The acceptance function

$$\alpha_{\text{MH}}(x^{(t)}, x^*) = \min \left\{ 1, \frac{\pi(x^*)}{\pi(x^{(t)})} \frac{q(x^{(t)} | x^*)}{q(x^* | x^{(t)})} \right\}, \quad (1)$$

is the MH acceptance function that keeps π invariant for this algorithm.

Algorithm 1 Metropolis–Hastings algorithm

```

1: Initialize: Set initial value  $x^{(0)}$ 
2: for  $t = 0, 1, 2, \dots, T - 1$  do
3:   Draw a proposal  $x^* \sim q(\cdot | x^{(t)})$ , where  $q$  is a proposal density.
4:   Draw  $W \sim \text{Uniform}(0, 1)$  independently.
5:   Compute the acceptance ratio:

$$r(x^{(t)}, x^*) = \frac{\pi(x^*)}{\pi(x^{(t)})} \frac{q(x^{(t)} | x^*)}{q(x^* | x^{(t)})}.$$

6:   if  $W \leq \min\{1, r(x^{(t)}, x^*)\}$  then
7:     Set  $x^{(t+1)} = x^*$ 
8:   else
9:     Set  $x^{(t+1)} = x^{(t)}$ 
10:  end if
11: end for
12: Return:  $\{x^{(t)}\}_{t=0}^{T-1}$ .
```

The effectiveness of an MH algorithm relies almost entirely on the choice of the proposal distribution, Q . A “good” proposal should often propose values in high probability areas of π and should also sufficiently explore the tails. Consequently, a “good” proposal

¹More recently, it has been highlighted that much of the laborious programming implementation in the original paper was carried out by Arianna Rosenbluth, and the name should possibly be changed to the Metropolis-Rosenbluth-Teller-Hastings Algorithm. Although we keep the common name in this paper, we direct the reader to Chen (2022), a wonderful article on the contributions of Arianna Rosenbluth.

makes large moves in the state-space, $\mathcal{X}$. A “bad” proposal is one that proposes values close to the current value, yielding low movement of the sampler.

Significant effort has gone into constructing effective sampling strategies. Perhaps the easiest one to implement is the original random walk Metropolis (RWM) of Metropolis *et al.* (1953) where $Q \equiv N(x^{(t)}, h)$ for a tunable $h > 0$. RWM algorithms are often a practitioner’s go-to choice, requiring very little overhead and guaranteeing quick implementation time. However, the proposal struggles to move rapidly in the state-space, particularly in high dimensions, as it utilizes no information from π itself on where high-probability areas might be. This led to the development of powerful gradient-based proposals like the Metropolis Adjusted Langevin Algorithm (MALA) (Roberts and Tweedie, 1996)

$$Q \equiv N\left(x^{(t)} + \frac{h}{2} \nabla \log \pi(x), h \mathbb{I}_d\right), \quad (2)$$

where $h > 0$ is tunable. The MALA proposal moves intentionally towards areas of high probability, learning from the local geometry. The result is an improved exploration of the state-space in high dimensions. HMC is also a gradient-based algorithm that cleverly uses multiple gradients to propose the next move of the sampler. However, HMC does not very elegantly fit into a typical MH scheme and can be viewed as an instance of the general Metropolis-Hastings-Green algorithm.

2.2. Metropolis-Hastings-Green algorithm

Consider a situation when we might want to employ a proposal that is *deterministic*. That is, there is nothing random about the proposal. Certainly, this will be extremely ineffective as the purpose of MCMC sampling is to obtain *random* draws from π . Green (1995), however, identified that one may augment the state-space in such a way that a deterministic proposal in an augmented state-space is in fact random in the original state-space. This is done through the Metropolis-Hastings-Green algorithm with Jacobians (MHGJ)².

Consider a conditional distribution $S(\cdot | x)$ with density $s(\cdot | x)$ defined on $\mathcal{Y}$ such that together with π , this yields the joint density

$$\pi_{\text{Aug}}(x, y) = \pi(x)s(y | x). \quad (3)$$

Since s is a valid density, integrating out y from π_{Aug} will yield our π of interest. In an MHGJ algorithm, the sampler will attempt to keep π_{Aug} invariant, however, we will only be retaining the x samples (in effect, integrating out y). The state-space for the augmented distribution is $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$. Typically, $S(\cdot | x)$ is chosen in such a way that sampling directly from it is straightforward.

The key idea in MHGJ is then the following: given a state $(x^{(t)}, y^{(t)})$, we first draw a fresh y from the true conditional distribution $S(\cdot | x^{(t)})$, and then apply a *deterministic* mapping on the *random* $(x^{(t)}, y)$, yielding (x^*, y^*) . This new state, (x^*, y^*) , is then accepted

²The Metropolis-Hastings-Green algorithm has found incredible use in transdimensional MCMC algorithms, and its connections to HMC are somewhat less known. Here, we present only what is essential for their use in HMC.

or rejected based on some acceptance probability that is designed to keep π_{Aug} invariant. Since we ignore the y samples, the marginal sampler will be π -invariant. The deterministic mapping here is a central piece and must satisfy certain properties. We denote this function by g and require that $g : \mathcal{Z} \rightarrow \mathcal{Z}$ satisfies $g(g(z)) = z$ for all $z \in \mathcal{Z}$ ³ and the Jacobian of g is everywhere non-singular. We present the MHGJ algorithm in Algorithm 2.

Algorithm 2 Metropolis–Hastings–Green with Jacobians (MHGJ) Algorithm

- 1: **Initialize:** Set initial value $x^{(0)}$
- 2: **for** $t = 0, 1, 2, \dots, T - 1$ **do**
- 3: Draw the auxiliary variable $y \sim S(\cdot | x^{(t)})$ with density $s(y | x^{(t)})$.
- 4: Apply the involution g , and obtain the proposed state

$$(x^*, y^*) = g(x^{(t)}, y).$$

- 5: Compute the acceptance ratio:

$$r(x^{(t)}, x^*) = \frac{\pi(x^*) s(y^* | x^*)}{\pi(x^{(t)}) s(y | x^{(t)})} \left| \det \left(\nabla g(x^{(t)}, y) \right) \right|.$$

- 6: Draw $W \sim \text{Uniform}(0, 1)$ independently.
 - 7: **if** $W \leq \min\{1, r(x^{(t)}, x^*)\}$ **then**
 - 8: Set $x^{(t+1)} = x^*$
 - 9: **else**
 - 10: Set $x^{(t+1)} = x^{(t)}$
 - 11: **end if**
 - 12: **end for**
 - 13: **Return:** $\{x^{(t)}\}_{t=0}^{T-1}$ as the Markov chain samples.
-

Algorithm 2 seems to employ an acceptance function noticeably different from standard MH. This is not entirely true, as we will now see. First, we note that Algorithm 2 should be viewed as a composition of two Markov kernels. The first kernel does a Gibbs step in Step 3, which keeps π_{Aug} invariant like any Gibbs full conditional update. The second kernel is comprised of Steps 4-11 for which the input is $(x^{(t)}, y)$. Now, given this input, Step 4 acts as a proposal step where the proposal is indeed *deterministic*. For Step 5, see that,

$$\begin{aligned} r(x^{(t)}, x^*) &= \frac{\pi(x^*) s(y^* | x^*)}{\pi(x^{(t)}) s(y | x^{(t)})} \left| \det \left(\nabla g(x^{(t)}, y) \right) \right| \\ &= \frac{\pi_{\text{Aug}}(x^*, y^*)}{\pi_{\text{Aug}}(x^{(t)}, y)} \left| \det \left(\nabla g(x^{(t)}, y) \right) \right|. \end{aligned} \tag{4}$$

This restructuring makes the acceptance function somewhat similar to (1), where the ratio of the target density at the proposed and the current points also appears. Since the proposal is a Dirac mass, only the Jacobian of the transformation remains in place of the ratio of the proposal densities in (1). See Tierney (1998) for more details on this last comment.

³Such a function is called an involution.

For the algorithm to be correct, g must be involutive, acting as its own inverse, ensuring reversibility of the deterministic proposal mechanism within the MHGJ framework. Glatt-Holtz *et al.* (2026); Neklyudov *et al.* (2020) present a holistic “involutive MCMC” paradigm, which shows that a wide class of MCMC kernels can be expressed via a self-inverse map g acting on an augmented state-space. Subsequent work has extended this idea to non-parametric models (Mak *et al.*, 2022), adaptive involutive map (Liu *et al.*, 2025), and to learned volume-preserving neural involutions (Spanbauer *et al.*, 2020).

Implementing the MHGJ algorithm thus entails specifying a suitable auxiliary distribution S and a deterministic map g . HMC does exactly this by choosing S and g by drawing connections with Hamiltonian dynamics. For most of Section 3, we assume the dimension $d = 1$ for ease of explanation. A general algorithm for a d -dimensional target is discussed in Section 3.3.2.

3. Hamiltonian Monte Carlo

To address computational challenges in lattice quantum chromodynamics, Duane *et al.* (1987) stitched molecular dynamics with MCMC, introducing the “Hybrid Monte Carlo” method. Since its introduction to statistics by Neal (1996) in neural network modeling, “Hybrid Monte Carlo” has become very popular and was later adopted as “Hamiltonian Monte Carlo” as a generic tool for sampling smooth distributions in Bayesian statistical inference. The Hamiltonian approach offers an alternative methodology for describing motion in particle systems and has served as the foundation for the development of HMC in the statistics community. We begin by first describing motion via Hamiltonian dynamics.

3.1. Hamiltonian dynamics

Imagine a ball in a large, frictionless one-dimensional bowl or valley. We may place the ball at any point in the bowl and give it a “push” to initiate its motion. Because the bowl is frictionless, the ball continues to move down and up the bowl according to its initial momentum. After descending into the bowl, it travels up the opposite side, gradually slowing as it climbs. At a sufficiently high point, the ball momentarily comes to rest before sliding back down the side of the bowl. The act of sliding back down corresponds to a change in the sign of the momentum. If the initial “push” is smaller, the highest point the ball can reach in the bowl is correspondingly lower. This behavior arises because the ball moves while conserving its total energy, which is determined jointly by the initial position and the momentum. As the ball moves up the sides of the bowl, its kinetic energy decreases (as its momentum reduces) while its potential energy increases (as its height increases), maintaining a constant total energy. A larger initial push gives the ball greater kinetic energy, allowing it to travel farther up the sides of the bowl.

The reader may further imagine that this system is simple enough that if a physicist were to close their eyes at some point while observing the ball, they would still be able to accurately predict the location of the ball after some time s . This is because the motion of the ball in the bowl follows well-defined laws that can be described using Hamiltonian dynamics. As we proceed in this section, we invite the reader to keep this ball-and-bowl analogy in mind to help build intuition.

Consider, now, the motion of an object on any frictionless surface. At any time t , let x_t denote the position of the particle and let p_t denote its momentum. At any given moment, an object possesses both potential energy and kinetic energy. The potential energy, $U(x_t)$, is determined by the position, and the kinetic energy, $K(p_t)$, is determined by its momentum. In the absence of external forces or friction, the object evolves in such a way that its total energy remains conserved. That is,

$$H(x_t, p_t) := U(x_t) + K(p_t)$$

remains constant as the object moves and is called the Hamiltonian. The actual value of the Hamiltonian is then dictated by the initial condition (x_0, p_0) , where the object started from and with what momentum. The conservation of the Hamiltonian is incredibly useful in determining how the object will move in a given system. Given U and K for a system, the temporal evolution of the object is governed by Hamilton's equations:

$$\frac{dx_t}{dt} = \frac{\partial H(x_t, p_t)}{\partial p_t} \quad \text{and} \quad \frac{dp_t}{dt} = -\frac{\partial H(x_t, p_t)}{\partial x_t}. \quad (5)$$

That is, (5) informs the change in the object's position and momentum with time. Given an (x_0, p_0) , if an object moves according to Hamilton's equations, then the Hamiltonian is conserved and $H(x_0, p_0) = H(x_t, p_t)$ for all t . Further, the solution to the partial differential equations in (5) allows for a functional relationship of the state of the object at any time t .

Running example 3.1. Consider a system where the potential, kinetic, and total (Hamiltonian) energies are expressed as

$$U(x) = \frac{x^2}{2}, \quad K(p) = \frac{p^2}{2}, \quad H(x, p) = \frac{x^2}{2} + \frac{p^2}{2}.$$

Applying Hamilton's equations, we obtain the system of differential equations

$$\frac{dx_t}{dt} = \frac{\partial H(x_t, p_t)}{\partial p_t} = p_t, \quad \frac{dp_t}{dt} = -\frac{\partial H(x_t, p_t)}{\partial x_t} = -x_t. \quad (6)$$

For initial condition (x_0, p_0) , the system of partial differential equations in (6) admits the analytical solution

$$x_t = x_0 \cos t + p_0 \sin t \quad \text{and} \quad p_t = -x_0 \sin t + p_0 \cos t. \quad (7)$$

An analytical solution, as in (7), allows the knowledge of the state of the object at any given time point t . This is eventually what will be used in the HMC sampler.

3.2. Ideal HMC

A key idea in HMC is to assume that the target density corresponds to a potential energy function. Specifically, we set

$$U(x) = -\log \pi(x).$$

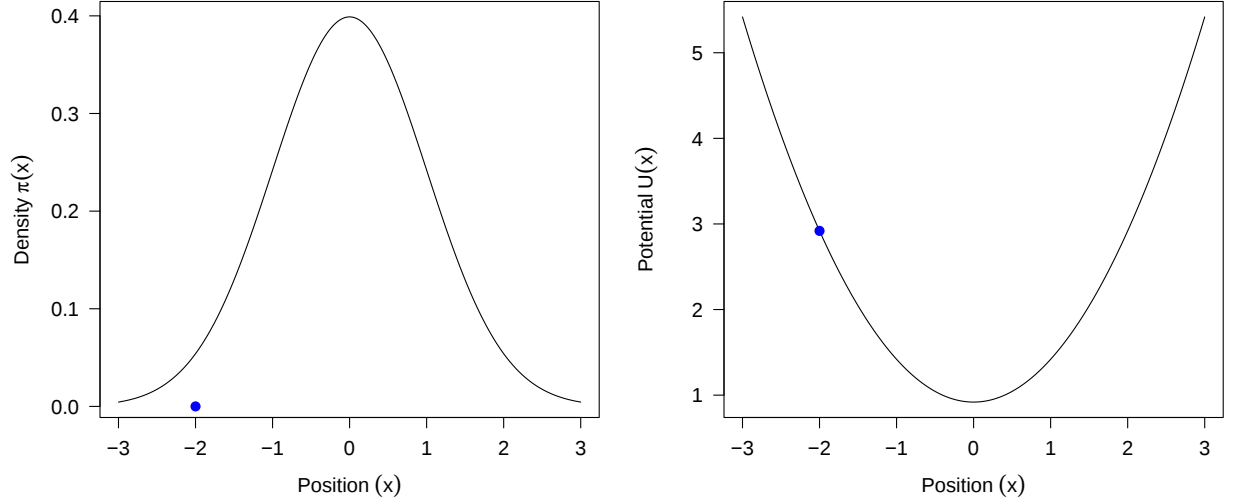


Figure 1: An equivalent representation of Gaussian potential energy.

Then a draw $X = x \sim \pi$ can be interpreted as a state possessing potential energy $U(x)$. Points located in the tails of the distribution correspond to regions of high potential energy, and points in the center of the distribution correspond to having low potential energy; see Figure 1. This might seem counterintuitive, but note that due to conservation of energy, high potential energy in the tails implies low kinetic energy. At a far enough point in the tail, the kinetic energy will be zero, at which point the direction of momentum will flip, making the object travel back to areas of high probability. If the object is assumed to have mass m , the kinetic energy function is

$$K(p) = \frac{p^2}{2m}.$$

Since we are setting up an imaginary system (there is no real object in motion), we may assume any form of kinetic energy; however, the above choice is standard.

Running example 3.2 (Normal distribution). Suppose $\pi \equiv N(0, 1)$, then $U(x) = x^2/2$ and Figure 1 plots this potential energy function. We may assume an associated kinetic energy of $K(p) = p^2/2$ so that $m = 1$. This yields the same Hamiltonian as in Example 3.1:

$$H(x, p) = \frac{x^2}{2} + \frac{p^2}{2}.$$

Given an initial condition (x_0, p_0) , the object will evolve according to the Hamiltonian dynamics (6) conserving energy in its path. Since an analytical solution is available in (7), we can draw the path of the object as time evolves, as a function of x and p . In Figure 2, the trajectory illustrates how the system evolves when initialized at $x_0 = 2$ and $p_0 = -2$. The particle begins its motion from this state and moves along a circular trajectory in the (x, p) plane, conserving its total energy throughout. Because the total energy is preserved for a given initial condition, different starting values of (x_0, p_0) correspond to distinct energy contours. In Figure 3, we consider a second trajectory with the same starting position, x_0 , but a different initial momentum, $p_0 = -1$. This can be thought of as putting an object at the same initial position but with a different initial momentum. In this case, the system

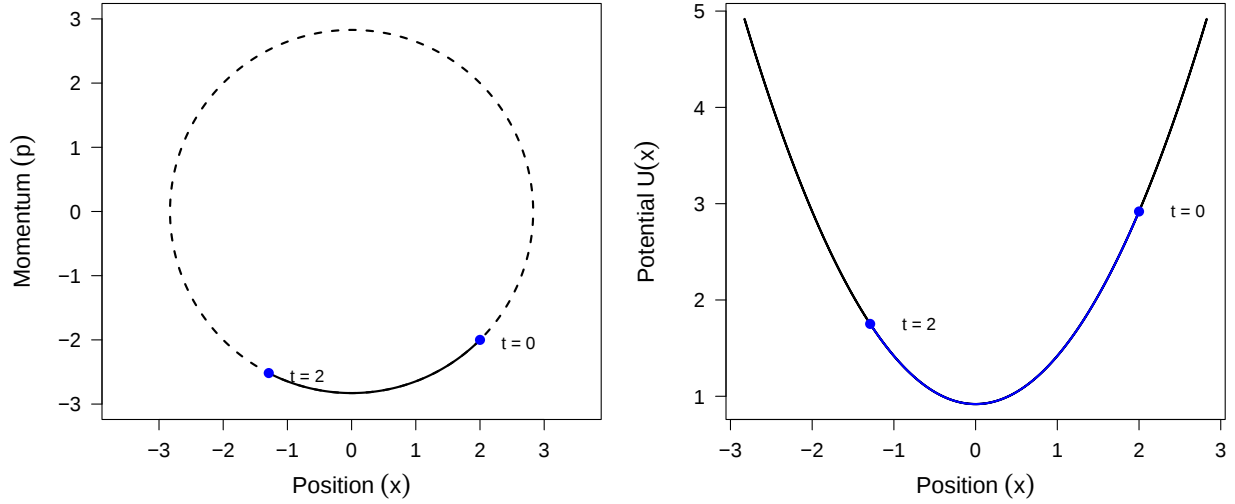


Figure 2: Hamiltonian dynamics under a Gaussian potential. The left panel depicts the trajectory of the particle in the (x, p) phase space, starting from $(x_0, p_0) = (2, -2)$ and its state after $t = 2$ time units. The right panel shows the potential energy function $U(x) = -\log \pi(x)$, illustrating the quadratic bowl corresponding to $\pi \equiv N(0, 1)$.

evolves along a separate energy contour.

We now present the ideal HMC algorithm, building upon the theoretical foundation of Hamiltonian dynamics established so far. The ideal HMC is essentially an MHGJ algorithm with a specific choice of auxiliary variable and involution. Here, the joint target density π_{Aug} is defined as

$$\pi_{\text{Aug}}(x, p) \propto e^{-H(x, p)} = e^{-U(x)} e^{-K(p)} \propto \pi(x) e^{-K(p)}. \quad (8)$$

The Hamiltonian curves in the (x, p) phase-space are the level curves of H , indicating points of equal value of the Hamiltonian; see the left plot of Figure 3. The ideal HMC sampler technically will sample from π_{Aug} , except the momentum samples are typically not saved. Notice here that by assumption, the marginal distribution of the momentum variable is $N(0, m)$, which is independent of the position.

The next ingredient to build an MHGJ sampler is to identify the involution; this is where the Hamiltonian dynamics are employed. Let $T_s(x_t, p_t)$ denote the progression of the object after s time units, starting from (x_t, p_t) . That is, let

$$(x_{t+s}, p_{t+s}) = T_s(x_t, p_t).$$

The map T_s has some nice properties. Namely,

1. If $s = 0$, the particle stays where it is. That is, $T_0(x_t, p_t) = (x_t, p_t)$.
2. The mapping is additive in time. That is, $T_{s_1+s_2}(x_t, p_t) = T_{s_1}(T_{s_2}(x_t, p_t)) = T_{s_2}(T_{s_1}(x_t, p_t))$.
3. The inverse of T_s is the mapping that goes back in time. That is, $T_s^{-1}(x_t, p_t) = T_{-s}(x_t, p_t)$.

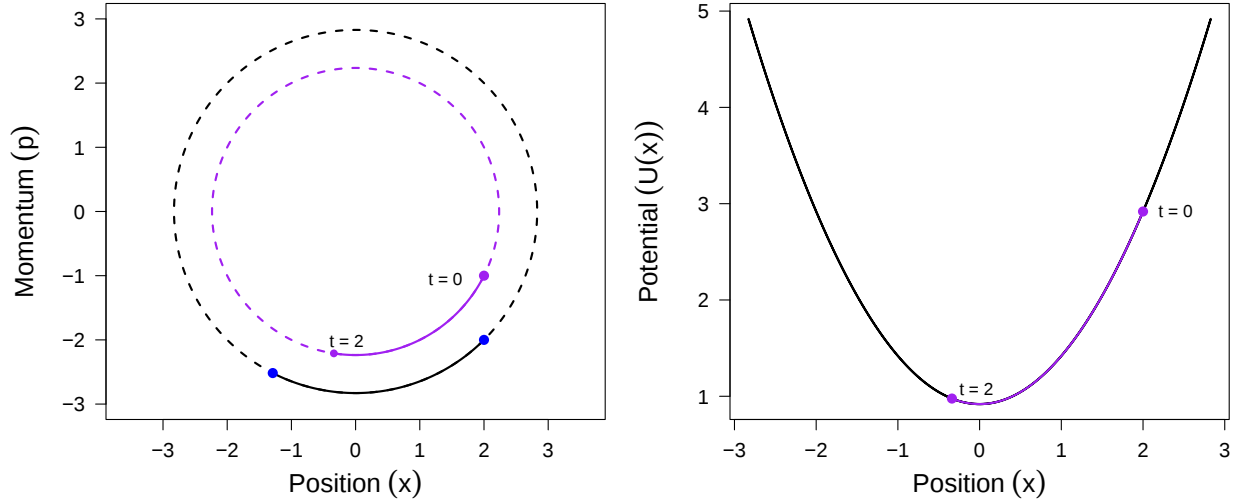


Figure 3: The left panel shows two distinct energy contours in the (x, p) space corresponding to different initial momenta. The right panel illustrates the particle's position after two time units for the second trajectory.

Despite these nice properties, the mapping T_s itself is not an involution and must be used in conjunction with another map. Define the map that reverses the momentum of the object as

$$i(x_t, p_t) := (x_t, -p_t),$$

called the momentum reversal map. Clearly, the mapping i is an involution. When the kinetic energy function is chosen to be symmetric about zero, that is $K(p) = K(-p)$, then together, T_s and i satisfy time-reversal symmetry which states that flipping the momentum of a system, moving the object s time units, and then flipping the momentum again is equivalent to going back in time s units. That is,

$$i \circ T_s \circ i(x_t, p_t) = T_{-s}(x_t, p_t).$$

Consider now the following composition map that pushes the object forward s time units and then reverses the momentum:

$$P_s(x_t, p_t) := i \circ T_s(x_t, p_t) = i(T_s(x_t, p_t)) = (x_{t+s}, -p_{t+s}). \quad (9)$$

The mapping P_s will be the final mapping that we will consider in the MHGJ algorithm. As it turns out, even though T_s is not an involution, P_s is. To see this, note that

$$\begin{aligned} P_s(P_s(x_t, p_t)) &= i(T_s(i(T_s(x_t, p_t)))) \\ &= T_{-s} \circ T_s(x_t, p_t) \\ &= T_s^{-1} \circ T_s(x_t, p_t) \\ &= (x_t, p_t). \end{aligned}$$

Thus, P_s is an involution. Moreover, since T_s is volume-preserving (Jordan, 2004, Liouville's theorem) and the Jacobian of the momentum flip i is a block diagonal matrix with

determinant ± 1 , therefore,

$$\left| \det \nabla P_s(x, p) \right| = 1, \quad (10)$$

showing that P_s is volume-preserving as well. Intuitively, volume-preservation implies that the volume of a region in the (x, p) space remains the same after P_s acts on it. Finally, since $K(p)$ is an even function of p , we have

$$H(x, -p) = U(x) + K(-p) = U(x) + K(p) = H(x, p),$$

ensuring energy invariance under momentum reversal. Using these ingredients, namely

- P_s is an involution,
- P_s is volume-preserving, and
- $K(p)$ is symmetric around zero,

we can run an MHGJ algorithm, yielding an ideal HMC algorithm in Algorithm 3. Specifically, Step 4 draws the auxiliary variable from its conditional distribution given the current iterate x ; in this instance, the conditional distribution is independent of x . In Step 5, we apply the involution P_s on (x, p) . Note here that this is akin to evolving the system using the Hamiltonian mapping T_s and then reversing the momentum. Since $K(p) = K(-p)$, reversing the momentum does not change the outcome of the algorithm and thus is not necessary during implementation. Step 6 calculates the acceptance ratio, which is deterministically 1, since in Step 5 (x^*, p^*) was obtained so as to conserve the Hamiltonian. Thus, the ideal HMC algorithm yields perfect acceptance and no rejections! Finally, since the momentum is resampled in Step 4, there is no need to save the momentum variables.

Algorithm 3 Ideal Hamiltonian Monte Carlo (HMC)

- 1: **Initialize:** Set initial value $x^{(0)}$.
- 2: **for** $t = 0, 1, 2, \dots, T - 1$ **do**
- 3: Given current state $x^{(t)}$.
- 4: Draw momentum $p \sim N(0, m)$.
- 5: Evolve the system for time s to obtain $(x^*, -p^*) = T_s(x^{(t)}, p)$.
- 6: Compute the Metropolis-Hastings ratio:

$$r(x^{(t)}, x^*) = e^{-H(x^*, p^*) + H(x^{(t)}, p)} = 1.$$

- 7: Set $x^{(t+1)} = x^*$.
 - 8: **end for**
 - 9: **Return:** $\{x^{(t)}\}_{t=0}^{T-1}$.
-

Perfect acceptance does not necessarily imply an excellent Markov chain. This is known for Gibbs samplers, which, although also having perfect acceptance, can exhibit slow mixing in various situations (Roberts and Sahu, 1997). The following running example will explain this point a bit more.

Running example 3.3. We continue with the example of $\pi \equiv N(0, 1)$ and $m = 1$ where recall

$$H(x, p) = \frac{x^2}{2} + \frac{p^2}{2}.$$

From (7), we have

$$T_s(x, p) = \begin{pmatrix} x \cos s + p \sin s \\ -x \sin s + p \cos s \end{pmatrix} \quad \text{and} \quad P_s(x, p) = \begin{pmatrix} x \cos s + p \sin s \\ x \sin s - p \cos s \end{pmatrix}.$$

Using the above involution P_s , the ideal HMC in Algorithm 3 can be implemented. Below is an R function that implements the ideal HMC for the standard Gaussian target.

```

1 # ideal HMC function
2 normal_idealHMC <- function(s = 1, n = 1e4)
3 {
4   x <- numeric(length = n)
5   x[1] <- 0 # starting at zero position
6   # don't need a starting value of p
7
8   for(k in 2:n)
9   {
10    p <- rnorm(1) # new momentum
11
12    # simulating Hamiltonian dynamics forward s time units
13    # no practical need to flip momentum
14    x[k] <- x[k-1]*cos(s) + p*sin(s)
15    p <- -x[k-1]*sin(s) + p*cos(s)
16  }
17  return(x) # not returning momentum
18 }
```

Figure 4 presents density estimates, trace plots, and autocorrelation function plots for three different choices of s . Noticeably, some choices of s are better than others, and perfect acceptance does not guarantee excellent mixing of the chain. In this implementation, $s = 1, 5$ yield well-behaved Markov chains, whereas $s = .1$ yields a significantly slower mixing Markov chain. This is because with $s = .1$, the object has not yet had the opportunity to travel far away from the current location, yielding smaller jump sizes, and thus higher autocorrelation.

The ideal HMC algorithm is called as such since, for most realistic examples, an analytical expression for T_s is not available. Thus, the involution cannot be applied directly. This roadblock leads to the widely accepted HMC algorithm, where the Hamiltonian dynamics are time-discretized to yield an approximation to T_s .

3.3. Hamiltonian Monte Carlo algorithm

In practical implementations, the map T_s is not available in closed-form, and Hamilton's equations must be approximated through numerical integration by discretizing time.

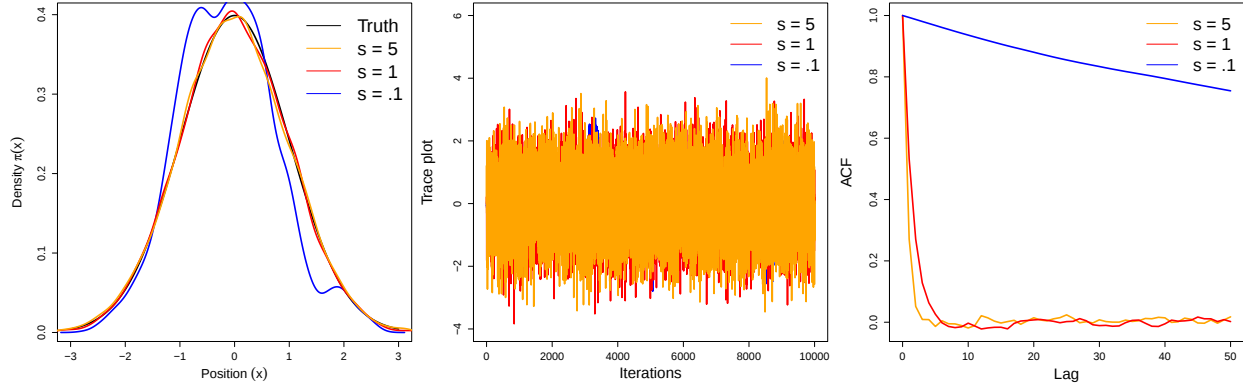


Figure 4: Density plots (left panel), trace plots (middle panel), and ACF plots (right panel) for ideal HMC on Gaussian target density with $s = 0.1, 1, 5$.

This yields an approximation to T_s that is used instead of T_s in Algorithm 3 to obtain the proposal x^* . Recall Hamilton's equations

$$\frac{dp_t}{dt} = -\frac{\partial H(x_t, p_t)}{\partial x_t} = -\nabla U(x_t), \quad \frac{dx_t}{dt} = \frac{\partial H(x_t, p_t)}{\partial p_t} = \nabla K(p_t) = \frac{p_t}{m}.$$

When a solution is analytically unavailable, a numerical integrator approximates dt with an $\varepsilon > 0$. Such a discretization moves the object approximately according to Hamilton's equations for $s = \varepsilon$ time units. As was seen in Figure 4, too small an s can yield a slow mixing Markov chain. Thus, such ε -step discretizations are applied L times so as to get $s = L\varepsilon$. However, any discretizer would only approximately be able to satisfy Hamilton's equations and thus would not be able to conserve the Hamiltonian exactly. Consider a standard Euler discretization:

$$p_{t+\varepsilon} = p_t - \varepsilon \nabla U(x_t) \quad \text{and} \quad x_{t+\varepsilon} = x_t + \varepsilon \nabla K(p_{t+\varepsilon}) = x_t + \varepsilon \frac{p_t}{m}. \quad (11)$$

As $\varepsilon \rightarrow 0$, consistent discretizers like the Euler discretization in (11) will yield improved conservation of the Hamiltonian. However, this does not guarantee long-time stability under repeated composition. That is, repeated application of ε -steps L times worsens the quality of the approximation. This has many consequences. First, any integrator will replace T_s with an approximation, $T_{L,\varepsilon}$ in Algorithm 3. In order for the resulting algorithm to still yield a π -invariant Markov chain, $T_{L,\varepsilon}$ and the resulting $P_{L,\varepsilon}$ must satisfy critical properties required by the MHGJ algorithm. That is, we require that

- $P_{L,\varepsilon}$ is also an involution, and
- $P_{L,\varepsilon}$ is also volume preserving.

The standard Euler discretizer violates both these properties. However, a popular discretizer that guarantees both these properties is the leapfrog integrator.

3.3.1. Leapfrog integrator

The leapfrog integrator introduces a staggered update sequence in which the momentum p is offset by $\varepsilon/2$ time units relative to the position x , yielding a scheme that forms the foundation of practical HMC implementations. Specifically, one jump of the leapfrog integrator first does an $\varepsilon/2$ -Euler jump for p , followed by an ε -Euler jump of x using the most updated state of p , with one last $\varepsilon/2$ -Euler jump of p . We can write a single leapfrog update over one step of size ε as:

1. Half-step update for momentum:

$$p_{t+\frac{\varepsilon}{2}} = p_t - \frac{\varepsilon}{2} \nabla U(x_t). \quad (12)$$

2. Full-step update for position:

$$x_{t+\varepsilon} = x_t + \varepsilon \frac{p_{t+\varepsilon/2}}{m}. \quad (13)$$

3. Final half-step update for momentum:

$$p_{t+\varepsilon} = p_{t+\frac{\varepsilon}{2}} - \frac{\varepsilon}{2} \nabla U(x_{t+\varepsilon}). \quad (14)$$

This procedure approximately advances the system by $s = \varepsilon$ time units. To simulate a longer trajectory of duration $s = L\varepsilon$, one simply performs L successive leapfrog steps, repeating (12)–(14). The final (ε, L) leapfrog integrator steps are

$$\begin{aligned} p_{t+\frac{\varepsilon}{2}} &= p_t - \frac{\varepsilon}{2} \nabla U(x_t), \\ x_{t+\varepsilon} &= x_t + \varepsilon \frac{p_{t+\varepsilon/2}}{m}, \\ p_{t+\frac{3\varepsilon}{2}} &= p_{t+\varepsilon/2} - \varepsilon \nabla U(x_{t+\varepsilon}), \\ &\vdots \\ x_{t+L\varepsilon} &= x_{t+(L-1)\varepsilon} + \varepsilon \frac{p_{t+(2L-1)\varepsilon/2}}{m}, \\ p_{t+L\varepsilon} &= p_{t+(2L-1)\varepsilon/2} - \frac{\varepsilon}{2} \nabla U(x_{t+L\varepsilon}). \end{aligned} \quad (15)$$

The leapfrog integrator is a *symplectic* integrator, implying both volume-preservation and the fact that repeated ε steps do not yield poorer approximations. This is evidenced in Figure 5 where $L = 20$ repeated ε -steps do not yield a systematic propagation of error. Further, it remains true that for $\varepsilon \approx 0$, the approximate path of the object stays closer to the true Hamiltonian than that for larger ε .

The leapfrog scheme in (15) thereby defines a mapping $T_{L,\varepsilon}(x, p)$ that approximates the exact Hamiltonian flow $T_s(x, p)$, where $s = L\varepsilon$. As before, we define

$$P_{L,\varepsilon}(x, p) = i(T_{L,\varepsilon}(x, p)), \quad \text{where recall } i(x, p) = (x, -p). \quad (16)$$

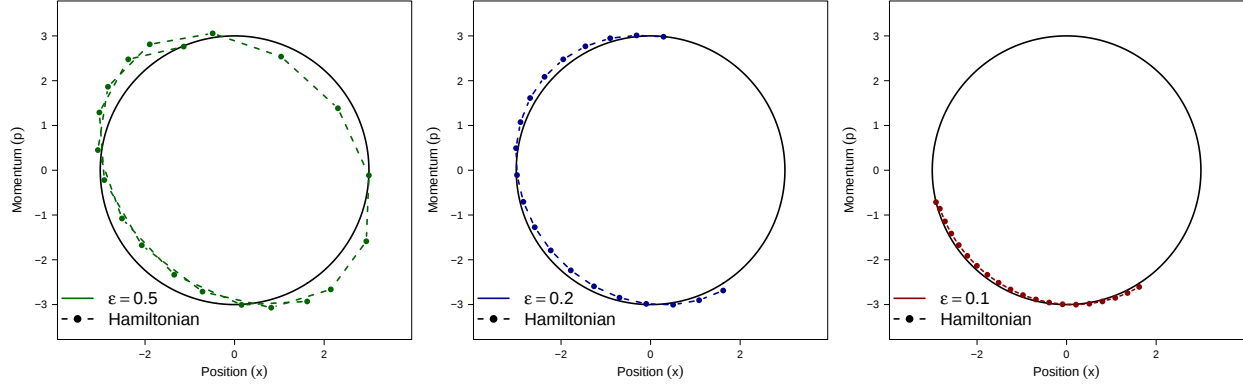


Figure 5: Approximated Hamiltonian paths with leapfrog discretization for different $\varepsilon = 0.5, 0.2, 0.1$ (from left to right) for standard Gaussian target distribution.

The function $P_{L,\varepsilon}$ in (16) is both an involution and volume-preserving due to the nice properties of the leapfrog integrator. However, the resulting $P_{L,\varepsilon}(x, p)$ does not conserve energy exactly, and this implies that the acceptance ratio need not be 1. As before, flipping the sign is not important in practice. The resulting implementable version of HMC using the leapfrog integrator is provided in Algorithm 4. This is the HMC algorithm, known and used widely in practice.

Algorithm 4 Hamiltonian Monte Carlo for 1-dimensional targets

- 1: **Initialize:** Set initial value $x^{(0)}$.
 - 2: **for** $t = 0, 1, 2, \dots, T - 1$ **do**
 - 3: Given current state $x^{(t)}$.
 - 4: Draw momentum $p \sim N(0, m)$.
 - 5: Compute $(x^*, -p^*) = T_{L,\varepsilon}(x^{(t)}, p)$ using L leapfrog steps of size ε .
 - 6: Compute the acceptance ratio:

$$r(x^{(t)}, x^*) = \exp\{-H(x^*, p^*) + H(x^{(t)}, p)\}.$$
 - 7: Draw $W \sim \text{Uniform}(0, 1)$, independently.
 - 8: **if** $W \leq \min\{1, r(x^{(t)}, x^*)\}$ **then**
 - 9: Set $x^{(t+1)} = x^*$.
 - 10: **else**
 - 11: Set $x^{(t+1)} = x^{(t)}$.
 - 12: **end if**
 - 13: **end for**
 - 14: **Return:** $\{x^{(t)}\}_{t=0}^{T-1}$.
-

Running example 3.4. For $\pi \equiv N(0, 1)$ and $m = 1$,

$$\nabla K(p) = p \text{ and } \nabla U(x) = x.$$

We implement Algorithm 4 in the R code below.

```

1 normal_HMC <- function(L = 10, eps = .1, n = 1e4)
2 {
3   x <- numeric(length = n)
4   x[1] <- 0 # starting from x = 0
5   # vectorizing this outside to save time in R
6   momentums <- rnorm(n)
7
8   for(k in 2:n)
9   {
10    p <- momentums[k] # new momentum
11    x_prop <- x[k-1]
12
13    # one half-Euler step
14    p_prop <- p - eps/2 * x_prop
15    # let the frog leap!
16    for(l in 1:L)
17    {
18      x_prop <- x_prop + eps * p_prop
19      if(l != L) p_prop <- p_prop - eps*x_prop
20    }
21    # one last half-Euler step
22    p_prop <- p_prop - eps/2 * x_prop
23
24    # Accept-reject on log scale
25    log.ratio <- (-x_prop^2 - p_prop^2 + x[k-1]^2 + p^2)/2
26    if(log(runif(1)) < log.ratio)
27    {
28      x[k] <- x_prop
29    } else{
30      x[k] <- x[k-1]
31    }
32  }
33  return(x)
34 }

```

We call the function with $s = 1$ for various choices of ε and present the results in Figure 6. The quality of the samples is fairly close to that of the ones obtained from the ideal HMC.

3.3.2. HMC in higher dimensions

So far, we have inherently assumed that π is defined on a 1-dimensional space. Naturally, this is rarely the case for the problems in which MCMC is employed. The extension to a general d -dimensional space is fairly straightforward. The potential $U(x)$ remains $U(x) = -\log \pi(x)$. However, the auxiliary variable (momentum) is also d -dimensional and for an appropriately chosen $p \times p$ mass matrix, M , the kinetic energy considered is

$$K(p) = \frac{1}{2}p^\top M^{-1}p.$$

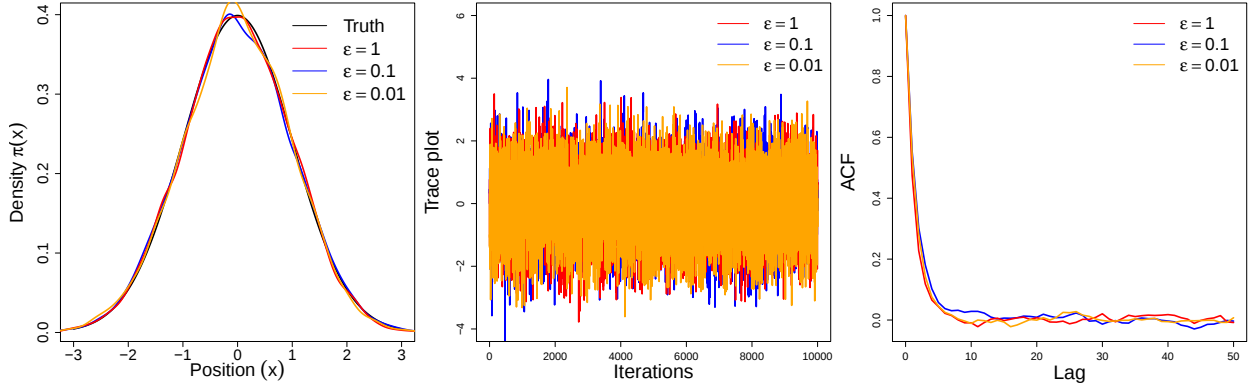


Figure 6: Density plots (left panel), trace plots (middle panel), and ACF plots (right panel) of HMC on Gaussian target density with different $\varepsilon = 0.01, 0.1, 1$ with $s = L\varepsilon = 1$.

Thus, the Hamiltonian on $\mathbb{R}^{2d}$ takes the form

$$H(x, p) = U(x) + K(p) = -\log \pi(x) + \frac{1}{2} p^\top M^{-1} p.$$

Consequently, $\nabla K(p) = M^{-1}p$. Given a step size $\varepsilon > 0$ and a number of steps L , the leapfrog in general dimensions can be written down as an extension of (12)–(14) as follows:

$$\begin{aligned} p_{t+\frac{\varepsilon}{2}} &= p_t - \frac{\varepsilon}{2} \nabla U(x_t), \\ x_{t+\varepsilon} &= x_t + \varepsilon M^{-1} p_{t+\varepsilon/2}, \\ p_{t+\frac{3\varepsilon}{2}} &= p_{t+\varepsilon/2} - \varepsilon \nabla U(x_{t+\varepsilon}), \\ &\vdots \\ x_{t+L\varepsilon} &= x_{t+(L-1)\varepsilon} + \varepsilon M^{-1} p_{t+(2L-1)\varepsilon/2}, \\ p_{t+L\varepsilon} &= p_{t+(2L-1)\varepsilon/2} - \frac{\varepsilon}{2} \nabla U(x_{t+L\varepsilon}). \end{aligned} \tag{17}$$

The final HMC algorithm in general dimensions is provided in Algorithm 5.

4. Choosing tuning parameters

A successful implementation of an HMC algorithm requires careful tuning of s, ε, L , and M . In this section, we provide recommendations for these parameters based on both heuristics and theory. Typically, a good strategy is to first choose M according to Section 4.1. Next, fix a small L and choose ε according to the guidelines in Section 4.2. Finally, using M and ε thus chosen, choose L and consequently $s = L\varepsilon$ based on the guidelines in Section 4.3.

Algorithm 5 Hamiltonian Monte Carlo for d -dimensional targets

```

1: Initialize: Set initial value  $x^{(0)}$ .
2: for  $t = 0, 1, 2, \dots, T - 1$  do
3:   Given current state  $x^{(t)}$ .
4:   Draw momentum  $p \sim N_d(0, M)$ .
5:   Compute  $(x^*, -p^*) = T_{L,\varepsilon}(x^{(t)}, p)$  using  $L$  leapfrog steps of size  $\varepsilon$  using (17).
6:   Compute the acceptance ratio:

$$r(x^{(t)}, x^*) = \exp\{-H(x^*, p^*) + H(x^{(t)}, p)\}.$$

7:   Draw  $W \sim \text{Uniform}(0, 1)$ , independently.
8:   if  $W \leq \min\{1, r(x^{(t)}, x^*)\}$  then
9:     Set  $x^{(t+1)} = x^*$ .
10:  else
11:    Set  $x^{(t+1)} = x^{(t)}$ .
12:  end if
13: end for
14: Return:  $\{x^{(t)}\}_{t=0}^{T-1}$ .
```

4.1. Mass matrix M

A first practical and natural choice for M is setting $M = \mathbb{I}_d$. Usually, a practitioner first tries this choice, and if the sampler is ineffective, a more informed choice of M is made. A practical workflow is demonstrated in Section 5. When a mass matrix different from $\mathbb{I}_d$ is employed, this is akin to linear preconditioning.

The choice of the mass matrix controls how the algorithm moves through the augmented parameter space. Intuitively, the mass matrix determines the relative speed at which the d different directions are explored during the simulated Hamiltonian trajectories. If these speeds are poorly matched to the scale of the target distribution, the resulting trajectories can become inefficient, requiring very small step sizes and leading to poor mixing. From a geometric perspective, a linear transformation of the position variables is equivalent to the inverse linear transformation of the momentum variables, and the mass matrix provides a convenient way to implement this re-scaling without explicitly reparametrizing the model. Considering this, an ideal choice of M is such that

$$M^{-1} = \text{Cov}_\pi(X) =: \Sigma.$$

However, Σ will typically be unavailable in practice, in which case, a warm-up chain is run from which Σ is estimated using a sample covariance matrix employing the sample mean, $\bar{x} = T^{-1} \sum_{t=0}^{T-1} x^{(t)}$:

$$\hat{\Sigma} = \frac{1}{T-1} \sum_{t=0}^{T-1} (x^{(t)} - \bar{x})(x^{(t)} - \bar{x})^\top. \quad (18)$$

Consequently, $M = \hat{\Sigma}^{-1}$ is chosen. There are two disadvantages of this method, (i) the estimator in (18) may be poor due to poor warm-up samples, and (ii) the sampler can be

expensive to implement due to the need of employing a matrix-vector multiplication for every leapfrog step.

A reasonable alternative is to employ a diagonal M with elements $1/\text{diag}(\hat{\Sigma})$. That is, the diagonals of M are set to be the marginal inverse sample variances for all components. This is a computationally cheaper alternative that can often work well, which we also employ in Section 5. However, recently, Hird and Livingstone (2025) show that it is in fact possible for this type of preconditioning to yield worse results than no preconditioning at all. They also propose a computationally viable alternate preconditioning method that proves to be more effective than diagonal preconditioning.

While an appropriate mass matrix can dramatically improve performance, it is important to recognize the limitations of linear preconditioning. Mass matrix adaptation is most effective when the target is well approximated by an elliptically contoured distribution, such as a multivariate Gaussian. For more complex targets exhibiting strong nonlinearity, varying curvature, or pronounced non-elliptical structure, no fixed linear transformation can simultaneously regularize the geometry everywhere. In such cases, Girolami and Calderhead (2011) propose Riemannian manifold HMC methods that incorporate position-dependent geometry where the mass matrix M is allowed to also depend on x . We expand on this a bit more in Section 6.3.

4.2. Discretization size ε

Having found an appropriate M , we move on to choosing ε . This parameter is more easily tunable due to the measurable effects and consequences of choosing a particular ε .

Recall that the acceptance probability for the ideal HMC in Algorithm 3 is exactly 1 due to the exact conservation of the Hamiltonian. In most situations, the solution to Hamilton's equations is unavailable, and the leapfrog integrator in Section 3.3.1 is used to simulate the Hamiltonian dynamics. This approximation leads to a change in the Hamiltonian, no longer ensuring an acceptance probability of exactly 1. Recall, however, that the Hamiltonian is increasingly better conserved as $\varepsilon \rightarrow 0$. Thus, the choice of the discretization step, ε , dictates the acceptance probability of the algorithm. A smaller value of ε will yield higher acceptance, and a larger value of ε will yield lower acceptance. Further, due to the characteristics of the leapfrog integrator, for a fixed ε , the acceptance probability is minimally affected by the choice of L .

Running example 4.1. For our running example, we implement the HMC algorithm for $s = L\varepsilon = 1$ and $s = L\varepsilon = 10$ in Table 1. We make two observations: first, the same value of ε for different values of L yields similar acceptances. This indicates that indeed the acceptance probability is primarily affected by ε . Second, acceptances decrease as ε increases. As always, large acceptance is not necessarily an indicator of a good Markov chain. In Figure 7, we present visual summaries of the HMC algorithm with $s = L\varepsilon = 10$. This figure is in stark contrast to Figure 6, where all choices of ε yielded well-behaved Markov chains. Here, when ε is too large, the chain gets stuck at the current position for a while since the Hamiltonian of the proposal deviates significantly from the Hamiltonian of the current value.

As discussed previously, a larger acceptance probability need not imply an excellent Markov chain. Using techniques from the optimal scaling literature (Roberts and Rosenthal,

Table 1: Acceptance rates for Leapfrog-based HMC algorithm with varying (L, ε) settings. Total trajectory length is $s = L\varepsilon$.

$s = L\varepsilon = 1$				$s = L\varepsilon = 10$			
Chain	ε	L	Acceptance	Chain	ε	L	Acceptance
1	0.01	100	1.0000	1	0.1	100	0.9995
2	0.1	10	0.9996	2	1	10	0.9155
3	1	1	0.9181	3	10	1	0.0037

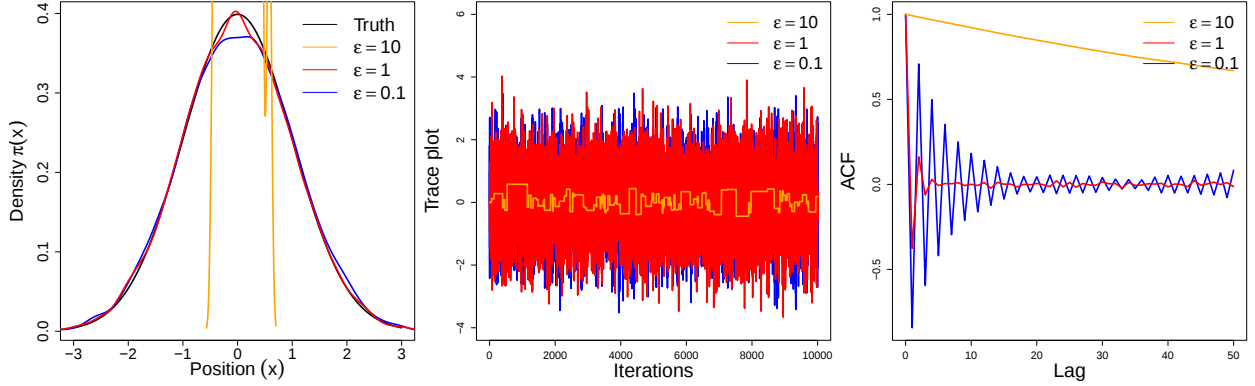


Figure 7: Density plots (left panel) and trace plots (right panel) for exact HMC on Gaussian target density with different $\varepsilon = 0.1, 1, 10$ with $s = L\varepsilon = 10$.

2001), Beskos *et al.* (2013) showed that as dimension $d \rightarrow \infty$, a user should tune ε to yield an acceptance rate of 0.651. However, for large but finite d , their simulations showed that higher acceptance rates up to 0.85 can yield better results. In general, users often choose ε aiming for 65% acceptance for complicated and high-dimensional target distributions and tune for acceptances closer to 85% for simpler targets. The tuning of ε can then be done using shorter length Markov chains than the final run, employing a starting value that is generally in an area of high probability.

4.3. Number of leapfrog steps and s

Having chosen M and ε , we move on to choosing the number of leapfrog steps, L . The performance of HMC depends critically on the number of leapfrog steps. Too few steps and the chain will explore the space like a slow random walk; too many, and we waste computation. A reasonable way then to choose L is to assess the computational budget. Each step of the leapfrog requires the calculation of a gradient vector, which is often the leading cost in the algorithm. Thus, L can be chosen based on the computation available. Note that for a given ε , a large L is unlikely to affect the acceptance probability of the HMC algorithm. Thus, if we have the computational bandwidth to increase L , it is often preferable.

There is, however, a limit to how large we might want L to be. It is possible for L to be large enough that a push of $s = L\varepsilon$ time units brings the object back close to the original position. This depends on the period of the Hamiltonian. Ideally, we would like to choose L so that $s = L\varepsilon$ is close to the half period of the underlying Hamiltonian. Then, the proposal

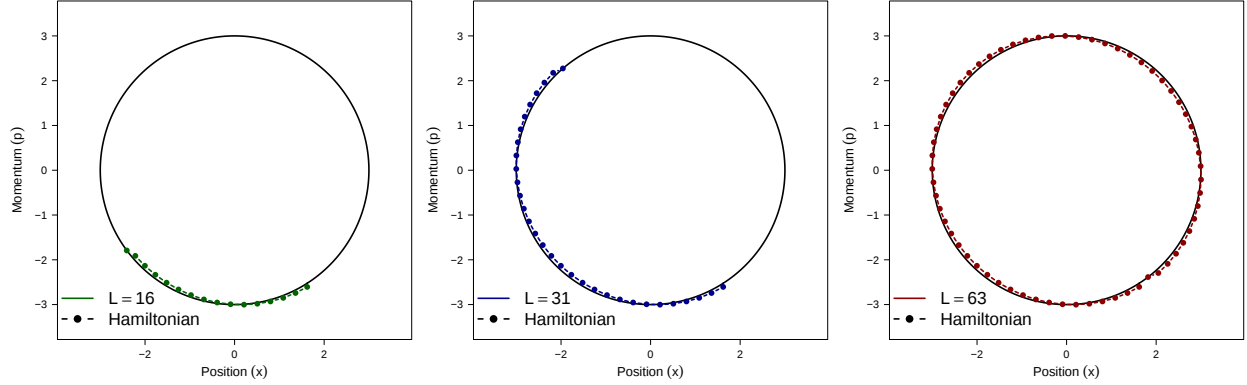


Figure 8: Approximated Hamiltonian paths with leapfrog discretization for fixed $\varepsilon = 0.1$ and with different L with time-trajectories $s = L\varepsilon$ at quarter period (left), half-period (middle), full-period (right) for standard Gaussian target distribution.

x^* is in general far from the current state of the object. This is challenging to do in practice since different parts of the state-space can have different periodicity. We discuss a popular adaptive procedure that attempts to solve this problem in Section 6.1.

Running example 4.2. In Figure 8, we demonstrate the quarter, half, and full period of the Gaussian Hamiltonian. For the same choice of ε (and thus same acceptance), each choice of L will yield different quality Markov chains. Increasing L beyond a value of 31 will be both wasteful and statistically inefficient.

In the next section, we discuss the implementation of HMC for a Bayesian logistic regression model on a real and popular dataset, demonstrating some key learnings from this section.

5. Example implementation

We present an HMC implementation for Bayesian logistic regression on the popular Pima Indian diabetes dataset (Chopin and Ridgway, 2017). This dataset was originally collected by the National Institute of Diabetes and Digestive and Kidney Diseases and contains information on adult female individuals of Pima Indian heritage, all aged at least 21 years and living near Phoenix, USA. Each response is a binary variable indicating the presence or absence of diabetes, together with covariates derived from medical examinations. We analyze a subset of the data available in the `MASS` package in `R`, consisting of $n = 200$ observations with no missing values. There are $d = 8$ covariates, including the intercept, number of pregnancies (`pregnant`), plasma glucose concentration measured two hours after an oral glucose tolerance test (`glucose`), diastolic blood pressure in mmHg (`pressure`), triceps skin-fold thickness in mm (`triceps`), body mass index (`mass`), defined as weight in kilograms divided by the square of height in meters, and the diabetes pedigree function (`pedigree`), which summarizes the genetic predisposition based on family history.

To study the relationship between these covariates and diabetes status, we employ a Bayesian logistic regression model of the form

$$Y_i \stackrel{\text{ind}}{\sim} \text{Bernoulli}(p_i) \text{ for all } i = 1, 2, \dots, n.$$

Denoting z_i as the covariate for the i th observation and the corresponding regression coefficient vector β , we model each p_i with the logit linked linear covariates as

$$\log\left(\frac{p_i}{1-p_i}\right) = z_i^\top \beta,$$

with independent priors on the regression coefficient $\beta \sim N_d(0, \sigma^2 \mathbb{I}_d)$. We choose the hyperparameter $\sigma^2 = 100$ to impose weak prior information. The resulting posterior is proper but intractable, and thus MCMC is used to obtain samples for posterior summaries.

First, we illustrate the importance of employing a suitable mass matrix as described in Section 4.1. We begin by running a baseline HMC algorithm using the identity mass matrix $M = \mathbb{I}_d$, following Algorithm 5. The sampler is run for 10^5 iterations with tuning parameters $L = 20$ and $\varepsilon = 2.1 \times 10^{-5}$, chosen in accordance with the guidelines in Section 4. The resulting trace plots shown in Figure 9 reveal that while most components of β exhibit satisfactory mixing, two components `Intercept` and `pedigree` mix slowly due to the variability of different scales present in the posterior. This lack of mixing leads to unreliable posterior density estimates, as displayed in Figure 10.

To mitigate the scale imbalance, we employ the diagonal preconditioning strategy described in Section 4.1. Specifically, we use a computationally cheaper diagonal mass matrix M , comprised of the inverse marginal sample variances estimated from warm-up iterations of a naive HMC run. We then rerun the HMC sampler for 10^5 iterations using the same leapfrog length $L = 20$, but with a substantially larger step size $\varepsilon = 0.11$, which maintains an acceptance rate consistent with the tuning considerations of Section 4. Figure 11 displays trace plots for the final 10^4 iterations of this preconditioned sampler, and the corresponding posterior density estimates are shown in Figure 12. Together, these results demonstrate a significant improvement in mixing across all components of β , as well as substantially more reliable posterior density estimation. The R codes to reproduce the simulation studies and real data analysis are available on GitHub⁴.

6. HMC variants

The potential power of HMC algorithms, as well as the need for its careful tuning, has led to several extensions and variations. We provide a short review of popular variants of the HMC algorithm proposed in the literature.

6.1. NUTS

Perhaps the most famous and powerful extension of the HMC algorithm is the No-U-Turn Sampler (NUTS). As seen in our previous sections, the efficiency of HMC critically depends on the tuning of the step size ε and the number of leapfrog steps L . If ε is too large relative to the local curvature of π , the numerical trajectory can diverge, while an excessively small ε leads to insufficient exploration. The NUTS (Hoffman and Gelman, 2014) eliminates the need to pre-specify the integration time $s = L\varepsilon$ by adaptively determining how long to simulate Hamiltonian dynamics. NUTS recursively constructs a trajectory by simulating Hamiltonian dynamics in both forward and backward time directions uniformly at random.

⁴Available at the repository https://github.com/ArghyaStat/HMC_for_dummies.

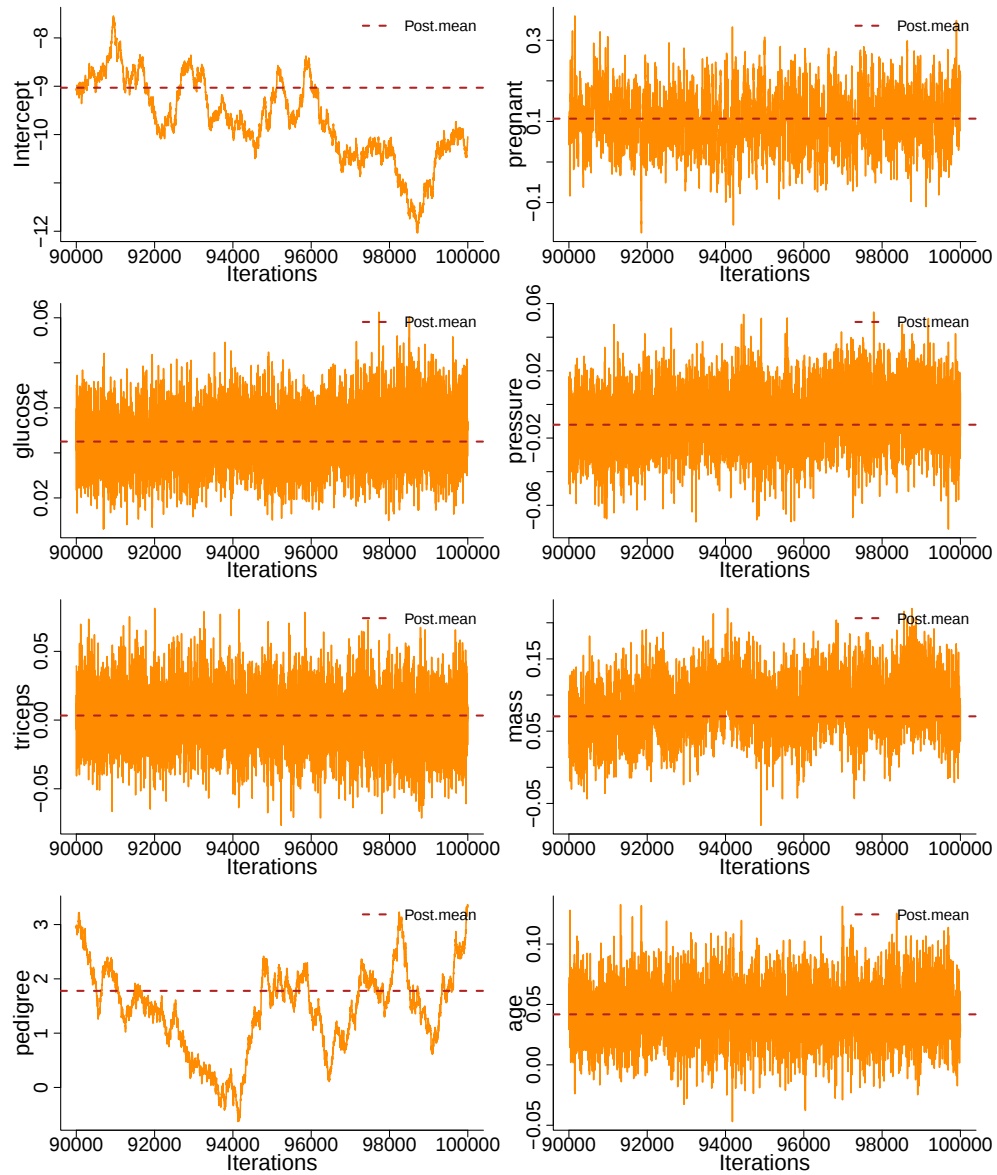


Figure 9: Traceplots of components of β with posterior mean (in dotted line) of the resulting Markov chain from the naive HMC sampler. The figure indicates that, except for Intercept and pedigree, all components are fast mixing.

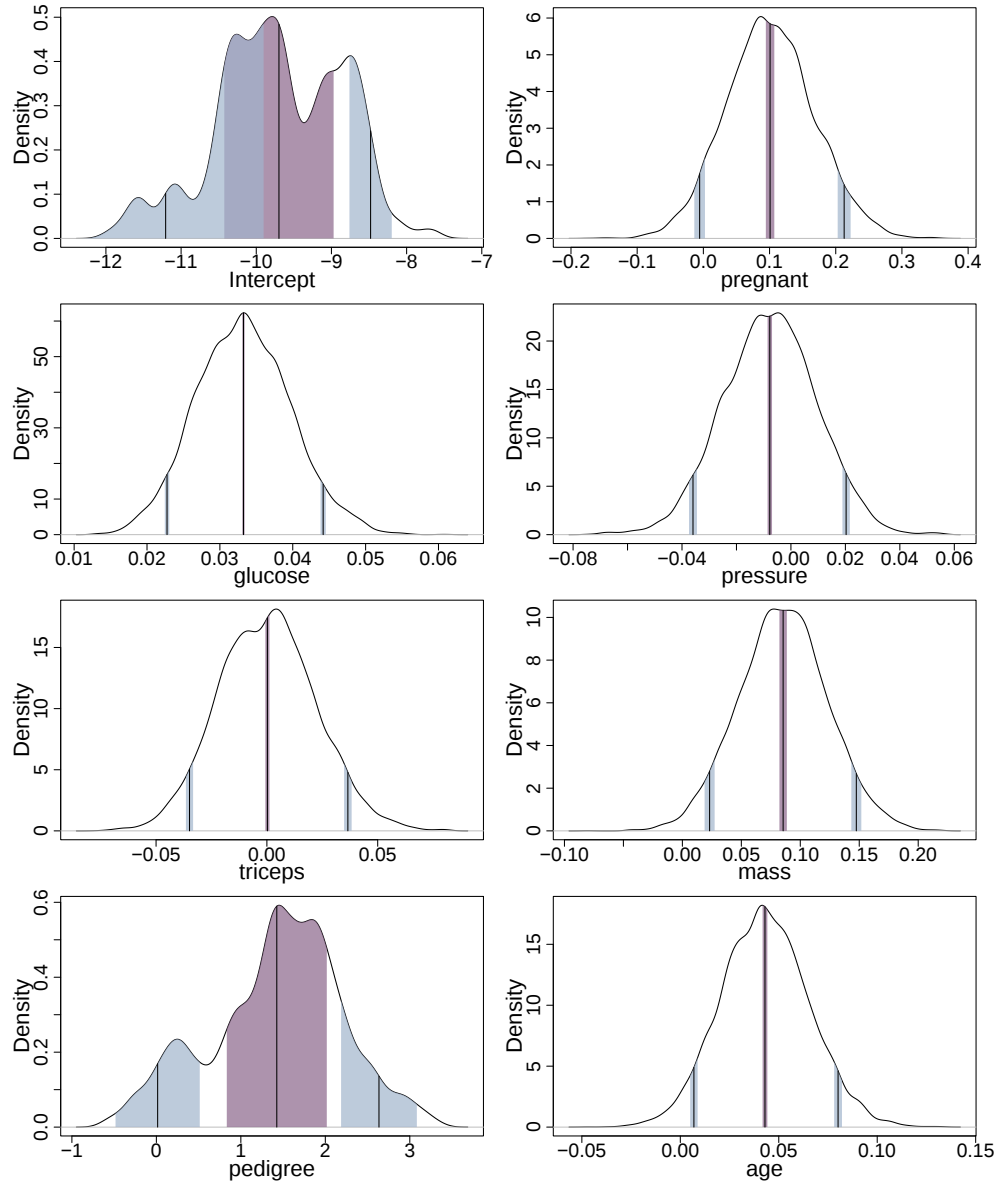


Figure 10: Density estimates for the components of β , with posterior means and the 5% and 95% estimated posterior quantiles from the naive HMC sampler. Bands around vertical lines show Monte Carlo uncertainty. Plot is made using R package SimTools.

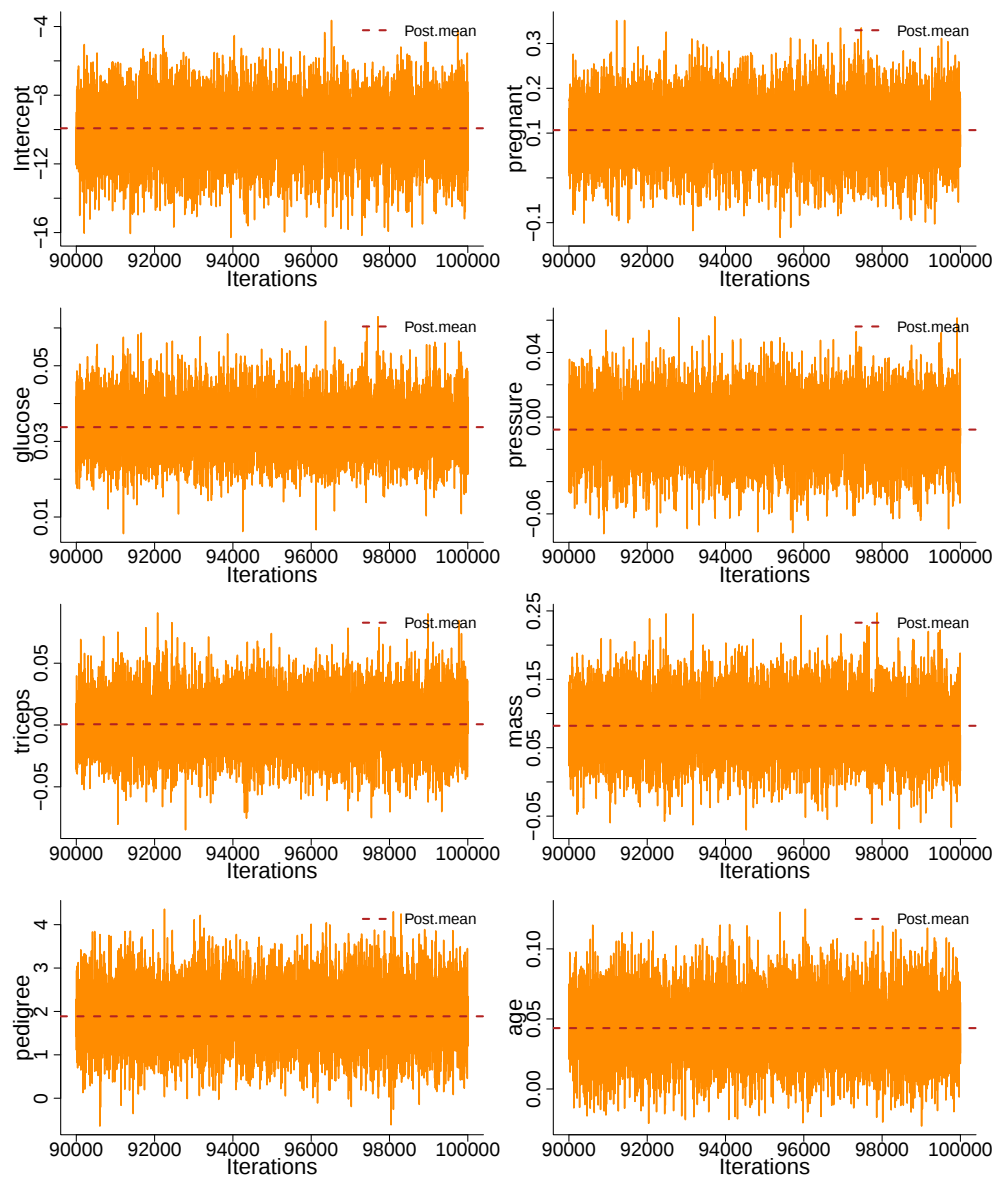


Figure 11: Trace plots of components of β with posterior mean (in dotted line).

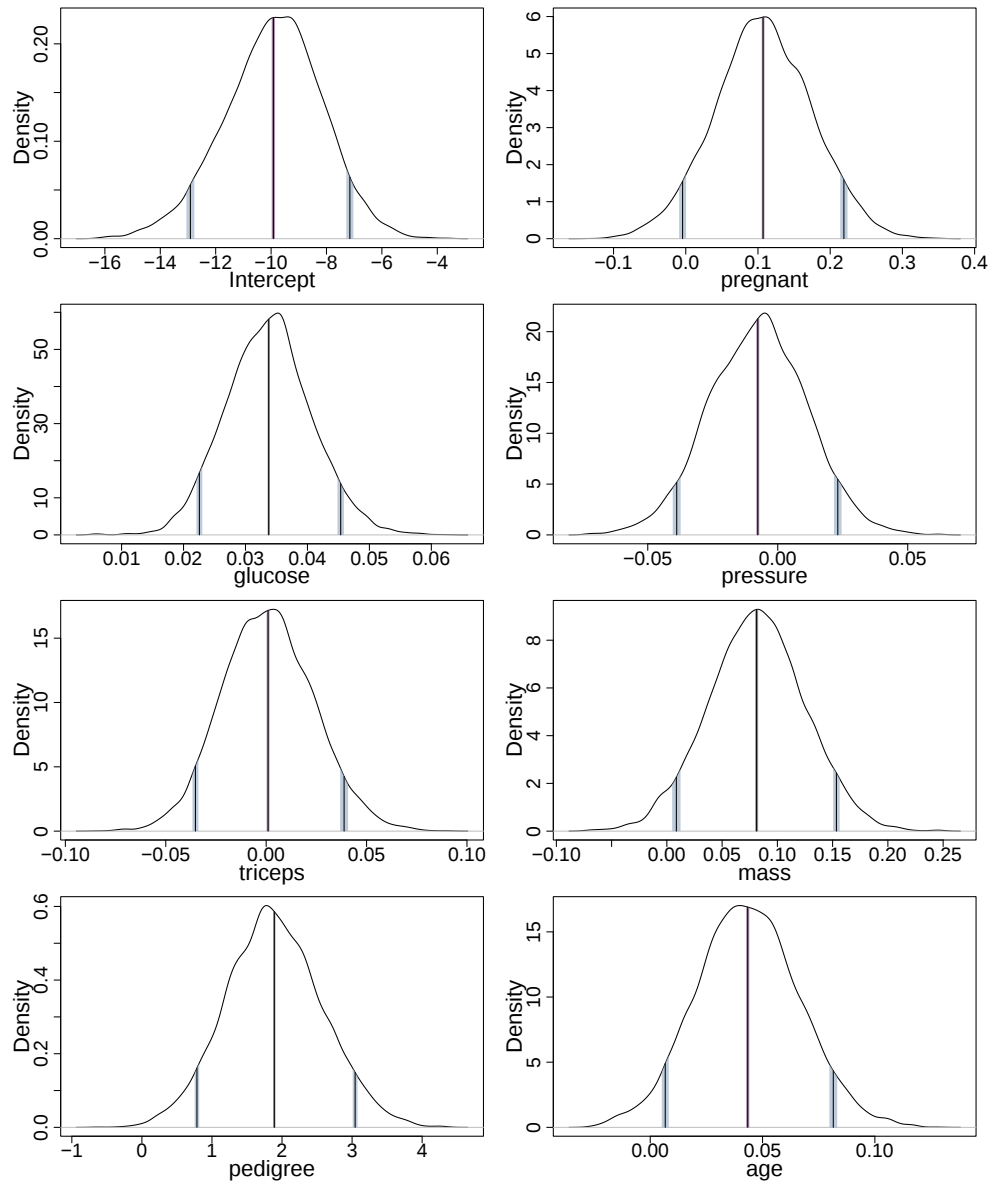


Figure 12: Density estimates for the components of β , with posterior means and the 5% and 95% estimated posterior quantiles for pre-conditioned sampler. The figure indicates that all components are well estimated.

The orbit is thus extended in that direction by doubling its length. The specifics of this ensure that the trajectory does not double back on itself. The resulting set of candidate states forms an orbit that is both symmetric and volume-preserving, thereby maintaining the reversibility of the Markov transition. From the constructed orbit, a single proposal (x^*, p^*) is drawn using a sampling scheme that preserves invariance. This adaptive construction enables NUTS to automatically adjust the integration time according to the local geometric features of the target distribution, substantially improving sampling efficiency without requiring manual tuning of L . Stan (Carpenter *et al.*, 2017) provides a widely used implementation of NUTS.

While NUTS adaptively addresses the integration length, it still employs a single global step size ε across the entire sampling process. In Stan, this choice of ε is determined in the warm-up phase using dual averaging (Nesterov, 2009) targeting desired acceptance rate. In multiscale problems, regions of high curvature in $U(x)$ may demand smaller step sizes for stability, whereas flatter regions permit larger ones. A fixed ε , therefore, presents a trade-off between numerical stability and efficiency. Bou-Rabee *et al.* (2025b) recently introduce orbit-level adaptive step-size variants of NUTS, which select ε separately for each orbit while maintaining detailed balance. Building further on this idea, Bou-Rabee *et al.* (2025a) propose the WALNUTS (Within-orbit Adaptive Leapfrog No-U-Turn Sampler), where the step size ε_t is allowed to vary within each orbit as a function of the local curvature, thereby ensuring stability in regions of high curvature while preserving efficiency in flatter regions. This localized adaptation improves robustness and energy control without violating reversibility or detailed balance.

6.2. Stochastic gradient HMC

One of the crucial bottlenecks of HMC is the need to compute the gradient of the log target repeatedly within a single iteration. Modern models frequently deal with datasets having a massive number of observations. In these ever-more-common scenarios of massive data, full gradient computation is infeasible. For example, suppose we want to sample from the posterior distribution of θ , the model parameter given a set of independent observations $\mathcal{D} = \{y_1, \dots, y_n\}$, denoted by

$$\pi(\theta|\mathcal{D}) = \left\{ \prod_{j=1}^n \log p(y_j|\theta) \right\} p(\theta) \propto \exp(-U(\theta)), \quad (19)$$

where $\log p(y_j|\theta)$ is the likelihood of θ based on y_j and $p(\theta)$ is the prior density of θ . The gradient of the potential energy function is thus given by

$$\nabla U(\theta) = - \sum_{j=1}^n \nabla \log p(y_j|\theta) - \nabla p(\theta). \quad (20)$$

Traditional stochastic methods (Welling and Teh, 2011) enjoy success by approximating $\nabla U(\theta)$ by $\nabla \tilde{U}(\theta)$ using minibatches of data, say $\tilde{\mathcal{D}}$, of size $k \ll n$ to scale algorithms for large n as

$$\nabla \tilde{U}(\theta) = - \frac{k}{n} \sum_{y_j: y_j \in \tilde{\mathcal{D}}} \nabla \log p(y_j|\theta) - \nabla p(\theta). \quad (21)$$

Chen *et al.* (2014) propose variants of stochastic gradient HMC with replacing ∇U with $\nabla \tilde{U}$. But HMC with stochastic gradient $\nabla \tilde{U}$ can remain costly due to the accept-reject step requiring the need to evaluate the full Hamiltonian. Chen *et al.* (2014) introduces a friction term in the Hamiltonian dynamics which allows the proposed sample path to be used without an accept-reject step. Note that the use of stochastic gradients comes at the cost of exactness of the algorithm.

Ma *et al.* (2017) provide a complete framework for stochastic gradient MCMC methods. They discussed a unified underlying stochastic dynamical system where HMC and SGHMC are special cases. Zou and Gu (2021) propose a stochastic variance-reduced HMC method for sampling from a smooth and strongly log-concave distribution. Their algorithm exhibits a faster rate of convergence and better gradient complexity than vanilla HMC and stochastic gradient HMC methods across a wide range of regimes. For a complete review of stochastic gradient MCMC methods, see Nemeth and Fearnhead (2021).

6.3. Riemannian manifold HMC

Sampling efficiency of HMC highly depends on the local geometric structure of the target distribution. Several extensions of HMC have sought to exploit this structure by modifying the kinetic energy function $K(p)$ through the introduction of global preconditioning matrices. A major advancement in this direction was proposed by Girolami and Calderhead (2011), who extended HMC from Euclidean manifolds to Riemannian manifolds by defining a position-dependent Gaussian momentum distribution. This is referred to as Riemannian manifold HMC (RHMC). In RHMC, the kinetic energy incorporates a second-order position-dependent metric tensor, typically the local Hessian of the potential function. Consequently, the Hamiltonian becomes non-separable in position (x) and momentum (p), rendering the generalized leapfrog updates computationally more expensive and often unstable compared to the true Hamiltonian trajectory.

Numerical instability arises because regions of high curvature in the parameter space can induce excessively large momentum realizations, leading to unstable trajectories unless a sufficiently small step size is employed. This restriction often necessitates conservative integration settings that limit efficiency across the parameter space. Inspired by the wide use of gradient clipping in deep learning, Lu *et al.* (2017) proposed a modified kinetic energy formulation that effectively bounds particle velocities during Hamiltonian evolution, mitigating the risk of instability. More recently, Whalley *et al.* (2024) investigated the use of randomized integration times (Bou-Rabee and Sanz-Serna, 2017) to enhance the numerical stability and mixing performance of RHMC. We discuss various numerical integrators and their stability properties in Section 6.6.

6.4. HMC for non-differentiable targets

Differentiability of the target density is a fundamental requirement for HMC. Nevertheless, Bayesian models formulated to induce structured sparsity frequently employ non-differentiable priors, thereby presenting substantial challenges in designing efficient sampling algorithms. Chaari *et al.* (2016) were among the first to address this issue by introducing a framework for sampling from probability distributions with non-differentiable energy functions, referred to as non-smooth HMC (ns-HMC). Their approach approximates the non-

differentiable potential using a smooth envelope. Although ns-HMC provides a principled solution to handle non-smooth potentials, its computational efficiency is limited when the proximal mapping of the potential is not available in closed form and must be evaluated through iterative optimization procedures.

More recently, Shukla *et al.* (2025) critically examined the limitations of ns-HMC and proposed the proximal HMC (p-HMC) algorithm, which integrates proximal mapping techniques within the Hamiltonian framework. They established theoretical guarantees of geometric ergodicity and offered practical guidelines for selecting tuning parameters associated with the proximal operator. Both ns-HMC and p-HMC essentially rely on smooth approximations of the non-smooth potential function via Moreau-Yosida envelopes for the purpose of gradient calculation. It remains to be seen whether there exist enveloping techniques that are particularly suited for Hamiltonian dynamics.

6.5. HMC for multi-modal distributions

Almost all standard MCMC methods struggle with crossing low-probability barriers that separate modes, and consequently take a long time to search for new modes or traverse from one mode to another, even in low dimensions. Łatuszyński *et al.* (2026) mention three major challenges of multi-modal sampling, namely: a) moving between the modes, b) finding the modes, and c) sampling efficiently within the modes. Gradient-based schemes, like HMC, particularly struggle with moving between modes since local gradients tend to push the sampler towards the current local basin of attraction.

The inability of HMC to sample effectively from multi-modal distributions has been widely acknowledged in the literature (Celeux *et al.*, 2000; Sminchisescu and Welling, 2007). For example, Mangoubi *et al.* (2018) show that for multi-modal target distributions, the ability of HMC to transition between modes is provably worse than that of random-walk Metropolis.

Tripuraneni *et al.* (2017) propose a non-canonical Hamiltonian dynamics called Magnetic HMC (MHMC), where the mechanics of the particle is coupled to a magnetic field. Magnetic dynamics accelerate the particle’s transition from one mode to another. Mongwe *et al.* (2021) proposed perturbed Hamiltonians designed with importance sampling. A more recent work in this context is Repelling-Attracting Hamiltonian Monte Carlo (RAHMC) by Vishwanath and Tak (2024). In RAHMC, the modified dynamics involve a friction parameter that allows (i) mode-repelling, that encourages the sampler to move away from regions of high probability density, and (ii) mode-attracting, that facilitates the sampler’s ability to find and settle near alternative modes. Moreover, Vishwanath and Tak (2024) show that the RAHMC preserves various important properties of traditional Hamiltonian dynamics like volume-preservation, time-reversibility, *etc.*

6.6. Numerical integrators for HMC

The main challenge in implementing the HMC method is generating the Hamiltonian trajectories themselves. Aside from a few trivial examples, we cannot solve Hamilton’s equations exactly, and any implementation must instead solve them numerically. Numerical inaccuracies, however, can quickly compromise the utility of even the most well-tuned

Hamiltonian transition. An active area of research is on using alternative integrators other than leapfrog integrators (Leimkuhler and Reich, 2004; Blanes *et al.*, 2014) in HMC. For highly curved posteriors, stability requires a small step size ε , which may lead to increased computational cost. To improve accuracy without reducing ε , higher-order symplectic methods (Yoshida, 1990) have been proposed, which are based on a symmetric composition of multiple leapfrog steps, achieving fourth-order accuracy. Brofos and Lederman (2021) study the implicit midpoint integrator in the context of RMHMC. They demonstrate that the implicit midpoint integrator exhibits superior conservation of energy, better preservation of phase-space volume and time-reversibility, compared to the standard generalized leapfrog in non-separable Hamiltonians.

Meanwhile, Pourzanjani and Petzold (2019) examines HMC for multiscale distributions where the standard leapfrog step size must be extremely small. They demonstrate that an implicit integrator enables larger step sizes and improved sampling efficiency in the presence of severe geometric non-linearity in the target. On the other hand, each step requires solving fixed-point or Newton iterations, which increases the per-step cost and complexity in implementation and tuning.

7. Discussion

The introduction of HMC has brought somewhat of a resurgence to Bayesian computation. Although not without its flaws, HMC has still been successfully employed in a wide variety of applications. Concurrently with methodological developments, there have been considerable efforts in also studying theoretical properties of HMC samplers. Durmus *et al.* (2017) were one of the first to study the ergodicity of the HMC algorithm. Livingstone *et al.* (2019) establish sufficient conditions for geometric ergodicity of HMC algorithms, with conditions similar to those required for MALA. Chen *et al.* (2020) study non-asymptotic bounds on the mixing time of HMC, quantifying the improvement compared to MALA.

More recently, together with stochastic and approximate sampling schemes, HMC has found great success in computation for machine learning. The use of HMC algorithms within the machine learning community has become prevalent, particularly for sampling from high-dimensional and potentially complicated distributions. We provide a brief review of the existing machine learning literature, where HMC plays a crucial role in sampling-based inference. Havasi *et al.* (2018) applies Stochastic Gradient HMC (SGHMC) for inference in deep Gaussian processes, demonstrating superior posterior estimation accuracy compared to the state-of-the-art doubly stochastic variational inference framework of Salimbeni and Deisenroth (2017). Their results illustrate the potential of Hamiltonian dynamics to provide uncertainty quantification beyond variational approximations.

Extending this direction, Foreman *et al.* (2021) propose the deep-learning HMC algorithm, a generalized formulation of HMC wherein a stack of neural network layers replaces traditional leapfrog updates. This approach enables the integration of deep learning architectures into the HMC framework, allowing adaptive and learnable transition dynamics. In the context of adversarial learning, Wang *et al.* (2022) introduce HMC with accumulated momentum (HMC-AM), a method that adaptively controls step sizes across the trajectory. By allowing particles to evolve with variable step sizes along different directions, HMC-AM enhances flexibility in navigating complex loss surfaces common in adversarial training.

Robnik *et al.* (2023) propose microcanonical HMC (MCHMC), which follows fixed energy Hamiltonian dynamics rather than the canonical ensemble used in standard HMC. Most recently, Lockwood *et al.* (2024) extend the classical HMC framework into the quantum domain by proposing quantum dynamical HMC (QDHMC). This hybrid approach leverages quantum computation as a proposal mechanism, replacing the classical symplectic integration step with simulations of quantum-coherent continuous-space dynamics on digital or analog quantum hardware. The QDHMC framework exemplifies an emerging intersection between quantum simulation and Bayesian computation, suggesting promising directions for scalable probabilistic inference in the quantum era.

Acknowledgments

The authors are thankful to Andrew Holbrook for his wonder talk available at <https://www.youtube.com/watch?v=Byr9JVBI9cUv> and some useful discussions that helped highlight the relationship between HMC and the MHGJ Algorithm. The authors are also thankful to Sameer Deshpande and Amy Herring for their encouragement to write this article. Finally, DV thanks Arushi Agarwal and Sarthak Parikh for creating a safe space to ask all the questions about Hamiltonian dynamics a “physics dummy” would ask.

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.
- Betancourt, M. (2017). A conceptual introduction to Hamiltonian Monte Carlo. *arXiv preprint arXiv:1701.02434*.
- Blanes, S., Casas, F., and Sanz-Serna, J. M. (2014). Numerical integrators for the hybrid Monte Carlo method. *SIAM Journal on Scientific Computing*, **36**, A1556–A1580.
- Bou-Rabee, N., Carpenter, B., Kleppe, T. S., and Liu, S. (2025a). The within-orbit Adaptive Leapfrog No-U-Turn sampler. *arXiv preprint arXiv:2506.18746*.
- Bou-Rabee, N., Carpenter, B., Kleppe, T. S., and Marsden, M. (2025b). Incorporating local step-size adaptivity into the No-U-Turn Sampler using Gibbs self-tuning. *The Journal of Chemical Physics*, **163**, 084119.
- Bou-Rabee, N. and Sanz-Serna, J. M. (2017). Randomized Hamiltonian Monte Carlo. *The Annals of Applied Probability*, **27**, 2159–2194.
- Brofos, J. and Lederman, R. R. (2021). Evaluating the implicit midpoint integrator for Riemannian Hamiltonian Monte Carlo. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 1072–1081. PMLR.
- Carpenter, B., Gelman, A., Hoffman, M. D., Lee, D., Goodrich, B., Betancourt, M., Brubaker, M., Guo, J., Li, P., and Riddell, A. (2017). Stan: A probabilistic programming language. *Journal of Statistical Software*, **76**, 1–32.
- Celeux, G., Hurn, M., and Robert, C. P. (2000). Computational and inferential difficulties with mixture posterior distributions. *Journal of the American Statistical Association*, **95**, 957–970.

- Chaari, L., Tournieret, J.-Y., Chaux, C., and Batatia, H. (2016). A Hamiltonian Monte Carlo method for non-smooth energy sampling. *IEEE Transactions on Signal Processing*, **64**, 5585–5594.
- Chen, S. (2022). Arianna Rosenbluth and the Metropolis Monte Carlo algorithm. *APS News*. <https://www.aps.org/apsnews/2022/03/rosenbluth-metropolis-monte-carlo-algorithm>.
- Chen, T., Fox, E., and Guestrin, C. (2014). Stochastic gradient Hamiltonian Monte Carlo. In *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 1683–1691. PMLR.
- Chen, Y., Dwivedi, R., Wainwright, M. J., and Yu, B. (2020). Fast mixing of Metropolized Hamiltonian Monte Carlo: Benefits of multi-step gradients. *Journal of Machine Learning Research*, **21**, 1–72.
- Chopin, N. and Ridgway, J. (2017). Leave Pima Indians alone: Binary regression as a benchmark for Bayesian computation. *Statistical Science*, **32**, 64–87.
- Duane, S., Kennedy, A. D., Pendleton, B. J., and Roweth, D. (1987). Hybrid Monte Carlo. *Physics letters B*, **195**, 216–222.
- Durmus, A., Moulines, E., and Saksman, E. (2017). On the convergence of Hamiltonian Monte Carlo. *arXiv preprint arXiv:1705.00166*.
- Foreman, S., Jin, X.-Y., and Osborn, J. C. (2021). Deep learning Hamiltonian Monte Carlo. *arXiv preprint arXiv:2105.03418*.
- Geman, S. and Geman, D. (1984). Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **PAMI-6**, 721–741.
- Girolami, M. and Calderhead, B. (2011). Riemann manifold Langevin and Hamiltonian Monte Carlo methods. *Journal of the Royal Statistical Society, Series B*, **73**, 123–214.
- Glatt-Holtz, N. E., Holbrook, A. J., Krometis, J. A., Mondaini, C. F., and Sheth, A. (2026). Sacred and profane: from the involutive theory of MCMC to helpful Hamiltonian hacks. In Craiu, R. V., Vats, D., Jones, G. L., Brooks, S., Gelman, A., and Meng, X.-L., editors, *Handbook of Markov Chain Monte Carlo*. Chapman & Hall, Boca Raton, 2nd edition.
- Granados, A. and Bañales, I. (2025). Understanding the Hamiltonian Monte Carlo through its physics fundamentals and examples. *arXiv preprint arXiv:2501.13932*.
- Green, P. J. (1995). Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrika*, **82**, 711–732.
- Hastings, W. (1970). Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, **57**, 97–109.
- Havasi, M., Hernández-Lobato, J. M., and Murillo-Fuentes, J. J. (2018). Inference in deep Gaussian processes using stochastic gradient Hamiltonian Monte Carlo. *Advances in Neural Information Processing Systems*, **31**.
- Hird, M. and Livingstone, S. (2025). Quantifying the effectiveness of linear preconditioning in Markov chain Monte Carlo. *Journal of Machine Learning Research*, **26**, 1–51.
- 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.

- Jordan, T. F. (2004). Steppingstones in Hamiltonian dynamics. *American Journal of Physics*, **72**, 1095–1099.
- Łatuszyński, K., Moores, M. T., and Stumpf-Fétizon, T. (2026). MCMC for multi-modal distributions. In Craiu, R. V., Vats, D., Jones, G. L., Brooks, S., Gelman, A., and Meng, X.-L., editors, *Handbook of Markov Chain Monte Carlo*. Chapman & Hall, Boca Raton, 2nd edition.
- Leimkuhler, B. and Reich, S. (2004). *Simulating Hamiltonian Dynamics*. Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press.
- Liu, T., Surjanovic, N., Biron-Lattes, M., Bouchard-Côté, A., and Campbell, T. (2025). Autostep: Locally adaptive involutive MCMC. In *Forty-second International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 39624–39650. PMLR.
- Livingstone, S., Betancourt, M., Byrne, S., and Girolami, M. (2019). On the geometric ergodicity of Hamiltonian Monte Carlo. *Bernoulli*, **25**, 3109–3138.
- Lockwood, O., Weiss, P., Aronshtein, F., and Verdon, G. (2024). Quantum dynamical Hamiltonian Monte Carlo. *Physical Review Research*, **6**, 033142.
- Lu, X., Perrone, V., Hasenclever, L., Teh, Y. W., and Vollmer, S. (2017). Relativistic Monte Carlo. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pages 1236–1245. PMLR.
- Ma, Y.-A., Foti, N. J., and Fox, E. B. (2017). Stochastic gradient MCMC methods for hidden Markov models. In *International Conference on Machine Learning*, pages 2265–2274. PMLR.
- Mak, C., Zaiser, F., and Ong, L. (2022). Nonparametric involutive Markov chain Monte Carlo. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 14802–14859. PMLR.
- Mangoubi, O., Pillai, N. S., and Smith, A. (2018). Does Hamiltonian Monte Carlo mix faster than a random walk on multimodal densities? *arXiv preprint arXiv:1808.03230*.
- Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E. (1953). Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, **21**, 1087–1092.
- Mongwe, W. T., Mbuva, R., and Marwala, T. (2021). Antithetic magnetic and shadow Hamiltonian Monte Carlo. *IEEE Access*, **9**, 49857–49867.
- Neal, R. M. (1996). Priors for infinite networks. In *Bayesian Learning for Neural Networks*, pages 29–53. Springer, New York.
- Neal, R. M. (2011). MCMC using Hamiltonian dynamics. In Brooks, S., Gelman, A., Jones, G. L., and Meng, X.-L., editors, *Handbook of Markov Chain Monte Carlo*, pages 113–162. Chapman and Hall/CRC, Boca Raton, FL.
- Neklyudov, K., Welling, M., Egorov, E., and Vetrov, D. (2020). Involutive MCMC: a unifying framework. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 7273–7282. PMLR.
- Nemeth, C. and Fearnhead, P. (2021). Stochastic gradient Markov chain Monte Carlo. *Journal of the American Statistical Association*, **116**, 433–450.
- Nesterov, Y. (2009). Primal-dual subgradient methods for convex problems. *Mathematical programming*, **120**, 221–259.

- Pourzanjani, A. A. and Petzold, L. R. (2019). Implicit Hamiltonian Monte Carlo for sampling multiscale distributions. *arXiv preprint arXiv:1911.05754*.
- Roberts, G. O. and Rosenthal, J. S. (2001). Optimal scaling for various Metropolis-Hastings algorithms. *Statistical Science*, **16**, 351–367.
- Roberts, G. O. and Sahu, S. K. (1997). Updating schemes, correlation structure, blocking and parametrization for the Gibbs sampler. *Journal of the Royal Statistical Society, Series B*, **59**, 291–317.
- Roberts, G. O. and Tweedie, R. L. (1996). Exponential convergence of Langevin distributions and their discrete approximations. *Bernoulli*, **2**, 341–363.
- Robnik, J., De Luca, G. B., Silverstein, E., and Seljak, U. (2023). Microcanonical Hamiltonian Monte Carlo. *Journal of Machine Learning Research*, **24**, 1–34.
- Salimbeni, H. and Deisenroth, M. (2017). Doubly stochastic variational inference for deep Gaussian processes. *Advances in Neural Information Processing Systems*, **30**.
- Salvatier, J., Wiecki, T. V., and Fonnesbeck, C. (2016). Probabilistic programming in Python using PyMC3. *PeerJ Computer Science*, **2**, e55.
- Shukla, A., Vats, D., and Chi, E. C. (2025). Proximal Hamiltonian Monte Carlo. *arXiv preprint arXiv:2510.22252*.
- Sminchisescu, C. and Welling, M. (2007). Generalized darting Monte Carlo. In *Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics*, volume 2 of *Proceedings of Machine Learning Research*, pages 516–523. PMLR.
- Spanbauer, S., Freer, C., and Mansinghka, V. (2020). Deep involutive generative models for neural MCMC. *arXiv preprint arXiv:2006.15167*.
- Thomas, S. and Tu, W. (2021). Learning Hamiltonian Monte carlo in R. *The American Statistician*, **75**, 403–413.
- Tierney, L. (1998). A note on Metropolis-Hastings kernels for general state spaces. *The Annals of Applied Probability*, **8**, 1–9.
- Tripuraneni, N., Rowland, M., Ghahramani, Z., and Turner, R. (2017). Magnetic Hamiltonian Monte Carlo. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 3453–3461. PMLR.
- Vishwanath, S. and Tak, H. (2024). Repelling-attracting Hamiltonian Monte Carlo. *arXiv preprint arXiv:2403.04607*.
- Wang, H., Li, G., Liu, X., and Lin, L. (2022). A Hamiltonian Monte Carlo method for probabilistic adversarial attack and learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **44**, 1725–1737.
- Welling, M. and Teh, Y. W. (2011). Bayesian learning via stochastic gradient Langevin dynamics. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 681–688.
- Whalley, P. A., Paulin, D., and Leimkuhler, B. (2024). Randomized time Riemannian manifold Hamiltonian Monte Carlo. *Statistics and Computing*, **34**, 48.
- Yoshida, H. (1990). Construction of higher order symplectic integrators. *Physics letters A*, **150**, 262–268.

Zou, D. and Gu, Q. (2021). On the convergence of Hamiltonian Monte Carlo with stochastic gradients. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 13012–13022. PMLR.