Introducing an Explicit Symplectic Integration Scheme for Riemannian Manifold Hamiltonian Monte Carlo

Adam D. Cobb
Department of Engineering Science
University of Oxford
acobb@robots.ox.ac.uk

Atılım Güneş Baydin
Department of Engineering Science
University of Oxford

Andrew Markham
Department of Computer Science
University of Oxford

Stephen J. Roberts
Department of Engineering Science
University of Oxford

Abstract

We introduce a recent symplectic integration scheme derived for solving physically motivated systems with non-separable Hamiltonians. We show its relevance to Riemannian manifold Hamiltonian Monte Carlo (RMHMC) and provide an alternative to the currently used generalised leapfrog symplectic integrator, which relies on solving multiple fixed point iterations to convergence. Via this approach, we are able to reduce the number of higher-order derivative calculations per leapfrog step. We explore the implications of this integrator and demonstrate its efficacy in reducing the computational burden of RMHMC. Our code is provided in a new open-source Python package, `hamiltorch`.

1 Introduction

In Bayesian statistics, we often find ourselves with the challenge of performing an integration over a set of model parameters $\boldsymbol{\omega}$. A common example is when we have a model, defined by the likelihood $p(\mathbf{y}|\mathbf{x}, \boldsymbol{\omega})$, with $\mathbf{x}$ and $\mathbf{y}$ constituting an input–output data pair. We define a reasonable prior over the model parameters, $p(\boldsymbol{\omega})$ and our interest lies in inferring the posterior over the parameters $p(\boldsymbol{\omega}|\mathbf{x}, \mathbf{y})$. Knowledge of this posterior allows us to make predictions over new input points $\mathbf{x}^*$ via the integration,

$$p(\mathbf{y}^*|\mathbf{x}^*, \mathbf{x}, \mathbf{y}) = \int p(\mathbf{y}^*|\mathbf{x}^*, \boldsymbol{\omega})\, p(\boldsymbol{\omega}|\mathbf{x}, \mathbf{y}) \mathrm{d}\boldsymbol{\omega}. \tag{1}$$

In Bayesian modelling, the path to this predictive distribution comes from employing Bayes' theorem:

$$p(\boldsymbol{\omega}|\mathbf{x}, \mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{x}, \boldsymbol{\omega})\, p(\boldsymbol{\omega})}{p(\mathbf{y}|\mathbf{x})}, \tag{2}$$

where the marginal likelihood, $p(\mathbf{y}|\mathbf{x})$, is the final term that needs to be inferred in order to obtain the predictive distribution in Equation (1). This marginal likelihood also requires integration with respect to $\boldsymbol{\omega}$:

$$p(\mathbf{y}|\mathbf{x}) = \int p(\mathbf{y}|\mathbf{x}, \boldsymbol{\omega})\, p(\boldsymbol{\omega}) \mathrm{d}\boldsymbol{\omega}. \tag{3}$$

Therefore, if we can infer $p(\mathbf{y}|\mathbf{x})$, we can evaluate the predictive distribution. Depending on the prior and the likelihood, there are sometimes analytic solutions to marginalising over $\boldsymbol{\omega}$, such as in Gaussian process regression [1], where the prior is conjugate to the likelihood. However, when this is

Preprint. Under review.

not the case, one must approximate the integral. One solution is to replace the true posterior with a variational approximation that is either cheaper to sample from (stochastic variational inference) [2] or conjugate to the likelihood [3]. Another option is to use a Markov chain Monte Carlo (MCMC) technique to sample directly from the posterior in Equation (2).

In this paper we focus on the latter option of using an MCMC technique, namely, Riemannian Manifold Hamiltonian Monte Carlo (RMHMC) [4]. As we describe in Sections 3 and 4, RMHMC is a Monte Carlo technique that is well suited to performing inference in highly complex models.

However, RMHMC implementations remain computationally expensive and challenging due to both the higher-order derivatives and quadratic memory requirements. It is for these reasons that RMHMC has not become widely used in large-scale Bayesian modelling, such as in Bayesian neural networks [3, 5]. To overcome these issues, we present a new RMHMC integration scheme, *Explicit RMHMC*, that is competitive with standard Hamiltonian Monte Carlo (HMC) in both wall-clock time and performance. Furthermore, we provide a Python package, `hamiltorch`, that implements a general-purpose API to run these algorithms on a GPU with PyTorch [6].[1]

Our paper is structured as follows. Section 2 reviews related literature, providing context for our work. Sections 3 and 4 provide theory, which are then followed by experimental results in Section 6. Finally, we provide closing remarks in Section 7.

2 Related work

Our motivation for introducing a more computationally efficient way of performing inference with RMHMC stems from quotes in the literature, such as *"we need improved inference techniques in BNNs"*, Gal and Smith [7]; and *"In small scale problems, Markov chain Monte Carlo algorithms remain the gold standard, but such algorithms face major problems in scaling up to big data."*, Li et al. [8]. Both these papers are referring to the need for improved inference in complex models. Furthermore they also highlight the drawbacks of variational approximations [9], which can lead to poor uncertainty quantification in safety-critical applications [10] and possibly be more susceptible to adversarial attacks [7].

2.1 Markov chain Monte Carlo methods

One of the most commonly used and most popularised MCMC algorithms originated from the statistical mechanics literature: the Metropolis–Hastings (MH) algorithm [11, 12] includes an acceptance step, whereby proposed samples are accepted according to an acceptance probability that ensures the required MCMC properties are satisfied.[2] Despite its success, the MH algorithm can suffer in high dimensional models due to low acceptance rates and an inability to efficiently explore the space (Bishop [13, Chapter 11]).

HMC has become a popular choice for performing inference in highly complex models, due to its higher acceptance rates and its use of gradient information in exploring the space. It was first introduced as "Hybrid Monte Carlo" by Duane et al. [14] for computer simulations in lattice field theory.[3] However the term "Hamiltonian Monte Carlo" was coined from the extensive work by Neal [16], who introduced HMC to the machine learning community with work on Bayesian neural networks. Despite the success of HMC, there are various hyperparameters that need to be tuned, such as the mass matrix, $\mathbf{M}$, the step size, ϵ, and the number of steps in a given trajectory, L (or leapfrog steps, see Section 3). The reliance on heuristics [16], which are heavily dependent on the Bayesian model or the data, often prevent HMC from being exploited to its full potential.

Progress in this area has been made with the introduction of a Hamiltonian-based Monte Carlo scheme that takes into account the geometry of the Bayesian model. This was framed in the work by Girolami

[1] Link to the Python package: `https://github.com/AdamCobb/hamiltorch`. For a tutorial please refer to `https://adamcobb.github.io/journal/hamiltorch.html`.

[2] This acceptance step was originally related to potential energy of particles in a system, such that a sample configuration of particles was accepted according to this potential energy, rather than the previous method of simply weighting samples by the energy distribution.

[3] We note that earlier work by Alder and Wainwright [15] utilised Hamiltonian dynamics to simulate molecular dynamics.

and Calderhead [4], where they introduced RMHMC as an *"overarching geometric framework"* for MCMC methods that included previous work by Roberts and Stramer [17] and Duane et al. [14].

2.2 Adaptations to Hamiltonian-based Monte Carlo samplers

The challenges associated with setting the hyperparameters of Hamiltonian samplers have seen various solutions offered in the literature. Hoffman and Gelman [18] focussed on solving the problem of automatically setting the trajectory length L in HMC by introducing the No-U-Turn Sampler, whilst ensuring detailed balance is preserved. This was extended to Riemannian manifolds in Betancourt [19] with the aim of reducing unnecessary computation time. Wang et al. [20] take a different approach to adapting hyperparameters by employing Bayesian optimisation [21] for adaptive tuning within Hamiltonian-based Monte Carlo samplers.

Rather than focussing on techniques for hyperparameter tuning, others have directly worked on the formulation of the Hamiltonian itself. Shahbaba et al. [22] show that it is possible to split the Hamiltonian in a way that can reduce the computational cost. They use a MAP approximation that allows the Hamiltonian to be partially analytically integrated, which can reduce the computational burden when exploring much of the state space. Previous work by Lan et al. [23] introduced some of the benefits of explicit integration techniques to RMHMC. However, unlike the integrator introduced in our work, they derive an integrator that is no longer symplectic and adjust their acceptance step in order to preserve detailed balance. In addition, they introduce two additional matrix inversions that limit efficiency in high-dimensional spaces.

3 Hamiltonian Monte Carlo

Rather than relying on a random walk to explore the parameter space of a model, HMC employs Hamiltonian dynamics to traverse the parameter space, $\boldsymbol{\omega} \in \mathbb{R}^D$. In order to introduce HMC, we begin with the integrand of interest $f(\boldsymbol{\omega}) = p(\mathbf{y}|\mathbf{x}, \boldsymbol{\omega})\, p(\boldsymbol{\omega})$ and augment the space by introducing the momentum variable $\mathbf{p} \in \mathbb{R}^D$, such that we now have a joint density $f(\boldsymbol{\omega})\, p(\mathbf{p}) = f(\boldsymbol{\omega})\, \mathcal{N}(\mathbf{p}|\mathbf{0}, \mathbf{M})$. The negative log joint probability is then given by

$$H(\boldsymbol{\omega}, \mathbf{p}) = -\log\left(f(\boldsymbol{\omega})\right) + \frac{1}{2}\log\left((2\pi)^D|\mathbf{M}|\right) + \frac{1}{2}\mathbf{p}^\top\mathbf{M}^{-1}\mathbf{p}. \tag{4}$$

It is then noted that this negative joint likelihood actually consists of a kinetic energy term $\frac{1}{2}\mathbf{p}^\top\mathbf{M}^{-1}\mathbf{p}$ and a potential energy term $-\mathcal{L}(\boldsymbol{\omega}) = -\log\left(f(\boldsymbol{\omega})\right)$. This insight points to using Hamiltonian dynamics to explore this system, where trajectories in this space move along constant energy levels.

Moving in this space requires derivative information:

$$\frac{\mathrm{d}\boldsymbol{\omega}}{\mathrm{d}\tau} = \frac{\partial H}{\partial \mathbf{p}} = \mathbf{M}^{-1}\mathbf{p}; \quad \frac{\mathrm{d}\boldsymbol{p}}{\mathrm{d}\tau} = -\frac{\partial H}{\partial \boldsymbol{\omega}} = \nabla_{\boldsymbol{\omega}}\mathcal{L}(\boldsymbol{\omega}). \tag{5}$$

The evolution of this system must preserve both volume and total energy, where any transformation that preserves area has the property of *symplecticity* [24, Page 182]. As this Hamiltonian is separable, meaning $\frac{\partial H}{\partial \mathbf{p}}$ and $\frac{\partial H}{\partial \boldsymbol{\omega}}$ are functions of only one of the variables (momentum and parameters, respectively), to traverse the space we can use the Stormer–Verlet or leapfrog integrator (as used in Duane et al. [14] and Neal [16]):

$$\mathbf{p}_{\tau+\epsilon/2} = \mathbf{p}_\tau + \frac{\epsilon}{2}\frac{\mathrm{d}\boldsymbol{p}}{\mathrm{d}\tau}(\boldsymbol{\omega}_\tau), \quad \boldsymbol{\omega}_{\tau+\epsilon} = \boldsymbol{\omega}_\tau + \epsilon\frac{\mathrm{d}\boldsymbol{\omega}}{\mathrm{d}\tau}(\mathbf{p}_{\tau+\epsilon/2}), \quad \mathbf{p}_{\tau+\epsilon} = \mathbf{p}_{\tau+\epsilon/2} + \frac{\epsilon}{2}\frac{\mathrm{d}\boldsymbol{p}}{\mathrm{d}\tau}(\boldsymbol{\omega}_{\tau+\epsilon})\ , \tag{6}$$

where τ corresponds to a the leapfrog step iteration and ϵ is the step size. Therefore given this is a numerical solution that uses discrete steps, we must check whether the Hamiltonian energy is conserved by employing a Metropolis–Hastings step with an acceptance probability $\min\{0, H_{\tau=0} - H_{\tau=L}\}$ after L leapfrog steps. Finally, the overall sampling process follows the Gibbs sampling structure, where we draw $\mathbf{p} \sim p(\mathbf{p})$ and then draw a new set of parameters $\boldsymbol{\omega}^* \sim p(\boldsymbol{\omega}^*|\mathbf{p}) \propto f(\boldsymbol{\omega}^*)\, p(\mathbf{p} + \delta\mathbf{p})$, where we have followed similar nomenclature to Girolami and Calderhead [4].

Therefore in repeating this Gibbs sampling scheme, including the implementation of the leapfrog algorithm, we can sample our parameter of interest $\boldsymbol{\omega}$.

4 Moving to a Riemannian manifold

One of the big challenges, when implementing HMC, is how to set the mass matrix $\mathbf{M}$. In Neal [16, Appendix 4], one practical solution is offered, which relates the step size for each parameter to the second-order derivative of the log posterior. Interestingly, this is not too far away from the topic of discussion in this section, where we start to explore ideas of higher-order derivatives.

If our interest lies in measuring distances between distributions, then naively treating the space as if it were Euclidean (while computationally convenient) may prevent rapid sampling of the posterior. A common metric for measuring a distance between distributions is the Kullback–Leibler (KL) divergence:

$$\mathrm{D}_{\mathrm{KL}}\big(p(\mathbf{x},\boldsymbol{\omega})||p(\mathbf{x},\boldsymbol{\omega}')\big) = \iint p(\mathbf{x},\boldsymbol{\omega})\left(\log p(\mathbf{x},\boldsymbol{\omega}) - \log p(\mathbf{x},\boldsymbol{\omega}')\right)\mathrm{d}\mathbf{x}\mathrm{d}\boldsymbol{\omega}. \tag{7}$$

If we expand the $\log p(\mathbf{x},\boldsymbol{\omega}')$ term within the integrand by using a Taylor expansion around $\boldsymbol{\omega}$ such that $\delta\boldsymbol{\omega} = \boldsymbol{\omega}' - \boldsymbol{\omega}$, we get (see Appendix A for full derivation)

$$\begin{aligned}\mathrm{D}_{\mathrm{KL}}\big(p(\mathbf{x},\boldsymbol{\omega})||p(\mathbf{x},\boldsymbol{\omega}+\delta\boldsymbol{\omega})\big) &\approx -\iint p(\mathbf{x},\boldsymbol{\omega})\left(\frac{1}{2}\delta\boldsymbol{\omega}^\top\nabla^2_{\boldsymbol{\omega}}\log p(\mathbf{x},\boldsymbol{\omega})\delta\boldsymbol{\omega}\right)\mathrm{d}\mathbf{x}\mathrm{d}\boldsymbol{\omega} \\ &= -\iint p(\mathbf{x},\boldsymbol{\omega})\left(\frac{1}{2}\delta\boldsymbol{\omega}^\top\mathbf{G}(\mathbf{x},\boldsymbol{\omega})\delta\boldsymbol{\omega}\right)\mathrm{d}\mathbf{x}\mathrm{d}\boldsymbol{\omega}\ ,\end{aligned} \tag{8}$$

where $\mathbf{G}(\mathbf{x},\boldsymbol{\omega})$ tells us about the curvature for the joint probability space over $\mathbf{x}$ and $\boldsymbol{\omega}$. As used in Girolami and Calderhead [4], we are using a Bayesian perspective and concern ourselves with the joint likelihood of the data and the parameters and therefore $\mathbf{G}(\boldsymbol{\omega})$ is the Fisher information plus the negative Hessian of the log-prior. Furthermore there is a plethora of literature relating to the appropriate metric for measuring distances on a Riemannian manifold, where we refer to either the RMHMC paper [4] or the book [25].

4.1 Riemannian manifold Hamiltonian Monte Carlo

Having introduced the need for a metric situated on a Riemannian manifold, we can now go back to the original Hamiltonian of Equation (4) but write it so that it lies on a Riemannian manifold by replacing $\mathbf{M}$ with $\mathbf{G}(\boldsymbol{\omega})$. However, because the mass matrix is now parameterised by $\boldsymbol{\omega}$, the Hamiltonian equations are no longer separable as they contain a coupling between $\boldsymbol{\omega}$ and $\mathbf{p}$:

$$\frac{\mathrm{d}\boldsymbol{\omega}}{\mathrm{d}\tau} = \frac{\partial H}{\partial \mathbf{p}} = \mathbf{G}(\boldsymbol{\omega})^{-1}\mathbf{p} \quad \text{and} \quad \frac{\mathrm{d}\boldsymbol{p}}{\mathrm{d}\tau} = -\frac{\partial H}{\partial \boldsymbol{\omega}} = \nabla_{\boldsymbol{\omega}}\mathcal{L}(\boldsymbol{\omega}) - \nabla_{\boldsymbol{\omega}}\mathcal{N}(\mathbf{0},\mathbf{G}(\boldsymbol{\omega})^{-1})\ . \tag{9}$$

Since the Hamiltonian is non-separable, the leapfrog integrator defined in Equation 6 is no longer applicable due to the volume conservation requirements. Furthermore, the new dependence of the mass matrix on the parameters means that detailed balance would no longer be satisfied.[4]

5 Explicit RMHMC

5.1 Implicit versus explicit integration

The current solution for solving the non-separable Hamiltonian equations in Section 4 is to rely on the *implicit* generalised leapfrog algorithm [26]. The term *implicit* refers to the computationally expensive first-order implicit Euler integrators, which are fixed-point iterations that are run until convergence. In order to understand the computational cost associated with this generalised leapfrog algorithm, we summarise a single leapfrog step in Algorithm 1.

A single step in the implicit generalised leapfrog contains two 'while' loops, which both require expensive update steps for each iteration. In Betancourt [27], the author uses 'while' loops as in Algorithm 1 and sets a threshold δ for breaking out. As an alternative to a conditional break, Girolami

[4] At this point, it feels important to highlight that in general, implementations of HMC are often not time-reversible, since that would require an equal chance of ϵ being positive or negative. However, as Hoffman and Gelman [18] mention, this can be omitted in practice.

and Calderhead [4] use a 'for' loop and fix the number of iterations to be five or six. Therefore, one step, (at a minimum) requires 11 Hessian calculations along with the additional derivative calculations.

Rather than relying on implicit integration, an *explicit* integration scheme relies on an explicit evaluation at an already known value [24, Chapter 1, Page 3]. This leads to a deterministic number of operations per leapfrog step, which we will show to be advantageous. We now introduce an explicit integration scheme for RMHMC.

5.2 Introducing the new explicit integrator

The challenge in speeding up the symplectic integration for non-separable Hamiltonians has not been applied within the context of Bayesian statistics. However if we delve into the physical sciences, there have been many examples where non-separable Hamiltonians appear [28–30]. Therefore it makes sense to draw from the progress made in this area, given the strong relationship between the physical interpretations of HMC within probabilistic modelling applications.

A key advance that moved away from dealing with specific subclasses of non-separable Hamiltonians and moved towards arbitrary non-separable Hamiltonians came in Pihajoki [31], where the phase space was extended to include an additional momentum term $\tilde{\mathbf{p}}$ and parameter term $\tilde{\boldsymbol{\omega}}$. This extension of the phase space was built on by Tao [32] who added an additional binding term that resulted in improved long-term performance. This new augmented Hamiltonian can now be defined as:

$$\tilde{H}(\boldsymbol{\omega}, \mathbf{p}, \tilde{\boldsymbol{\omega}}, \tilde{\mathbf{p}}) = H_1(\boldsymbol{\omega}, \tilde{\mathbf{p}}) + H_2(\tilde{\boldsymbol{\omega}}, \mathbf{p}) + \Omega h(\boldsymbol{\omega}, \mathbf{p}, \tilde{\boldsymbol{\omega}}, \tilde{\mathbf{p}}) \ , \tag{10}$$

where the binding term $h(\boldsymbol{\omega}, \mathbf{p}, \tilde{\boldsymbol{\omega}}, \tilde{\mathbf{p}}) = \|\boldsymbol{\omega} - \tilde{\boldsymbol{\omega}}\|_2^2/2 + \|\mathbf{p} - \tilde{\mathbf{p}}\|_2^2/2$ and Ω controls the binding. The two Hamiltonian systems H_1 and H_2 each follow Equation (4), where $H(\boldsymbol{\omega}, \tilde{\mathbf{p}})$ and $H(\tilde{\boldsymbol{\omega}}, \mathbf{p})$ have their parameters and momenta mixed. Interestingly, if we set $\Omega = 0$, we retrieve the augmented Hamiltonian from Pihajoki [31].

The Hamiltonian in Equation (10) gives the system:

$$\begin{aligned}
\dot{\boldsymbol{\omega}} &= \ \ \delta \partial_{\mathbf{p}} H(\tilde{\boldsymbol{\omega}}, \mathbf{p}) + \Omega(\mathbf{p} - \tilde{\mathbf{p}}) \\
\dot{\mathbf{p}} &= -\delta \partial_{\boldsymbol{\omega}} H(\boldsymbol{\omega}, \tilde{\mathbf{p}}) - \Omega(\boldsymbol{\omega} - \tilde{\boldsymbol{\omega}}) \\
\dot{\tilde{\boldsymbol{\omega}}} &= \ \ \delta \partial_{\tilde{\mathbf{p}}} H(\boldsymbol{\omega}, \tilde{\mathbf{p}}) + \Omega(\tilde{\mathbf{p}} - \mathbf{p}) \\
\dot{\tilde{\mathbf{p}}} &= -\delta \partial_{\tilde{\boldsymbol{\omega}}} H(\tilde{\boldsymbol{\omega}}, \mathbf{p}) - \Omega(\tilde{\boldsymbol{\omega}} - \boldsymbol{\omega}).
\end{aligned} \tag{11}$$

We now introduce the integrators from Tao [32], which when combined make up the symplectic flow for the augmented non-separable Hamiltonian:

$$\phi^{\delta}_{H_1} : \begin{bmatrix} \boldsymbol{\omega} \\ \mathbf{p} \\ \tilde{\boldsymbol{\omega}} \\ \tilde{\mathbf{p}} \end{bmatrix} \mapsto \begin{bmatrix} \boldsymbol{\omega} \\ \mathbf{p} - \delta \partial_{\boldsymbol{\omega}} H(\boldsymbol{\omega}, \tilde{\mathbf{p}}) \\ \tilde{\boldsymbol{\omega}} + \delta \partial_{\tilde{\mathbf{p}}} H(\boldsymbol{\omega}, \tilde{\mathbf{p}}) \\ \tilde{\mathbf{p}} \end{bmatrix}, \quad \phi^{\delta}_{H_2} : \begin{bmatrix} \boldsymbol{\omega} \\ \mathbf{p} \\ \tilde{\boldsymbol{\omega}} \\ \tilde{\mathbf{p}} \end{bmatrix} \mapsto \begin{bmatrix} \boldsymbol{\omega} + \delta \partial_{\mathbf{p}} H(\tilde{\boldsymbol{\omega}}, \mathbf{p}) \\ \mathbf{p} \\ \tilde{\boldsymbol{\omega}} \\ \tilde{\mathbf{p}} - \delta \partial_{\tilde{\boldsymbol{\omega}}} H(\tilde{\boldsymbol{\omega}}, \mathbf{p}) \end{bmatrix},$$

$$\phi^{\delta}_{\Omega h} : \begin{bmatrix} \boldsymbol{\omega} \\ \mathbf{p} \\ \tilde{\boldsymbol{\omega}} \\ \tilde{\mathbf{p}} \end{bmatrix} \mapsto \frac{1}{2} \begin{bmatrix} \begin{pmatrix} \boldsymbol{\omega} + \tilde{\boldsymbol{\omega}} \\ \mathbf{p} + \tilde{\mathbf{p}} \end{pmatrix} + \mathbf{R}(\delta) \begin{pmatrix} \boldsymbol{\omega} - \tilde{\boldsymbol{\omega}} \\ \mathbf{p} - \tilde{\mathbf{p}} \end{pmatrix} \\ \begin{pmatrix} \boldsymbol{\omega} + \tilde{\boldsymbol{\omega}} \\ \mathbf{p} + \tilde{\mathbf{p}} \end{pmatrix} - \mathbf{R}(\delta) \begin{pmatrix} \boldsymbol{\omega} - \tilde{\boldsymbol{\omega}} \\ \mathbf{p} - \tilde{\mathbf{p}} \end{pmatrix} \end{bmatrix}, \quad \text{where } \mathbf{R}(\delta) = \begin{bmatrix} \cos(2\Omega\delta)\mathbf{I} & \sin(2\Omega\delta)\mathbf{I} \\ -\sin(2\Omega\delta)\mathbf{I} & \cos(2\Omega\delta)\mathbf{I} \end{bmatrix}. \tag{12}$$

Finally, the numerical symplectic second-order integrator is given by the function composition

$$\phi^{\delta}_{\tilde{H}} = \phi^{\delta/2}_{H_1} \circ \phi^{\delta/2}_{H_2} \circ \phi^{\delta}_{\Omega h} \circ \phi^{\delta/2}_{H_2} \circ \phi^{\delta/2}_{H_1}, \tag{13}$$

where combining these symmetric mappings is known as Strang splitting [33] and higher-order integrators can be built in a similar manner [34]. In Appendix B, we show that this second-order integrator is indeed symplectic.

There are two major advantages when comparing the integrator in Equation (13) to previous symplectic integrators for non-separable Hamiltonians. First of all, when comparing to the implicit integrator of Section 5.1, we no longer need to rely on two fixed-point methods that can often diverge. Furthermore we are guaranteed the same number of derivative calculations of eight per equivalent

leapfrog step. This compares to the minimum number of 11 for the implicit generalised leapfrog (although it can often be higher). Secondly, when comparing to the previous explicit integrator of Pihajoki [31], this integration scheme has better long-term error performance. More specifically, Tao [32] showed that until the number of steps, $N = \mathcal{O}(\min(\delta^{-2}\Omega^{-1}, \Omega^{1/2}))$, the numerical error is $\mathcal{O}(N\delta^2\Omega)$, although empirical results show that larger values of N give favourable results in terms of the sampler's performance.

Algorithm 1 Implicit Leapfrog Step

Inputs: $\mathbf{p}_0, \boldsymbol{\omega}_0, \epsilon,$
$\mathbf{p} = \mathbf{p}_0$
while $\Delta p > \delta$ **do**
  $\mathbf{p}' = \mathbf{p}_0 + \frac{\epsilon}{2}\frac{\mathrm{d}\mathbf{p}}{\mathrm{d}\tau}(\mathbf{p}, \boldsymbol{\omega}_0)$
  $\Delta p = \max_i\{|p_i - p_i'|\}$
  $\mathbf{p} = \mathbf{p}'$
end while
$\boldsymbol{\omega} = \boldsymbol{\omega}_0$
while $\Delta\omega > \delta$ **do**
  $\boldsymbol{\omega}' = \boldsymbol{\omega}_0 + \frac{\epsilon}{2}\frac{\mathrm{d}\boldsymbol{\omega}}{\mathrm{d}\tau}(\mathbf{p}, \boldsymbol{\omega}_0) + \frac{\epsilon}{2}\frac{\mathrm{d}\boldsymbol{\omega}}{\mathrm{d}\tau}(\boldsymbol{p}, \boldsymbol{\omega})$
  $\Delta\omega = \max_i\{|\omega_i - \omega_i'|\}$
  $\boldsymbol{\omega} = \boldsymbol{\omega}'$
end while
$\mathbf{p} = \mathbf{p} + \frac{\epsilon}{2}\frac{\mathrm{d}\mathbf{p}}{\mathrm{d}\tau}(\mathbf{p}, \boldsymbol{\omega})$

Algorithm 2 Explicit Leapfrog Step

Inputs: $\mathbf{p}, \boldsymbol{\omega}, \tilde{\mathbf{p}}, \tilde{\boldsymbol{\omega}}, \epsilon, \Omega$
$\mathbf{p} = \mathbf{p} - \frac{\epsilon}{2}\partial_{\boldsymbol{\omega}}H(\boldsymbol{\omega}, \tilde{\mathbf{p}})$
$\tilde{\boldsymbol{\omega}} = \tilde{\boldsymbol{\omega}} + \frac{\epsilon}{2}\partial_{\tilde{\mathbf{p}}}H(\boldsymbol{\omega}, \tilde{\mathbf{p}})$
$\tilde{\mathbf{p}} = \tilde{\mathbf{p}} - \frac{\epsilon}{2}\partial_{\tilde{\boldsymbol{\omega}}}H(\tilde{\boldsymbol{\omega}}, \mathbf{p})$
$\boldsymbol{\omega} = \boldsymbol{\omega} + \frac{\epsilon}{2}\partial_{\mathbf{p}}H(\tilde{\boldsymbol{\omega}}, \mathbf{p})$
$c = \cos(2\Omega\epsilon), \quad s = \sin(2\Omega\epsilon)$
$\boldsymbol{\omega} = (\boldsymbol{\omega} + \tilde{\boldsymbol{\omega}} + c(\boldsymbol{\omega} - \tilde{\boldsymbol{\omega}}) + s(\mathbf{p} - \tilde{\mathbf{p}}))/2$
$\mathbf{p} = (\mathbf{p} + \tilde{\mathbf{p}} - s(\boldsymbol{\omega} - \tilde{\boldsymbol{\omega}}) + c(\mathbf{p} - \tilde{\mathbf{p}}))/2$
$\tilde{\boldsymbol{\omega}} = (\boldsymbol{\omega} + \tilde{\boldsymbol{\omega}} - c(\boldsymbol{\omega} - \tilde{\boldsymbol{\omega}}) - s(\mathbf{p} - \tilde{\mathbf{p}}))/2$
$\tilde{\mathbf{p}} = (\mathbf{p} + \tilde{\mathbf{p}} + s(\boldsymbol{\omega} - \tilde{\boldsymbol{\omega}}) - c(\mathbf{p} - \tilde{\mathbf{p}}))/2$
$\tilde{\mathbf{p}} = \tilde{\mathbf{p}} - \frac{\epsilon}{2}\partial_{\tilde{\boldsymbol{\omega}}}H(\tilde{\boldsymbol{\omega}}, \mathbf{p})$
$\boldsymbol{\omega} = \boldsymbol{\omega} + \frac{\epsilon}{2}\partial_{\mathbf{p}}H(\tilde{\boldsymbol{\omega}}, \mathbf{p})$
$\mathbf{p} = \mathbf{p} - \frac{\epsilon}{2}\partial_{\boldsymbol{\omega}}H(\boldsymbol{\omega}, \tilde{\mathbf{p}})$
$\tilde{\boldsymbol{\omega}} = \tilde{\boldsymbol{\omega}} + \frac{\epsilon}{2}\partial_{\tilde{\mathbf{p}}}H(\boldsymbol{\omega}, \tilde{\mathbf{p}})$

6 Experiments

6.1 Bayesian logistic regression

The purpose of this illustrative example is to demonstrate both the benefits of RMHMC, and that Explicit RMHMC is significantly faster than Implicit RMHMC. We start with the convex problem of Bayesian logistic regression as the metric tensor is positive semi-definitive and hence well behaved. We define our Hamiltonian as in Equation 4, where the log joint probability is given by

$$\mathcal{L}(\boldsymbol{\omega}) = \sum_i^N \{y_i \log(\boldsymbol{\omega}^\top \mathbf{x}_i) + (1 - y_i)\log(1 - \boldsymbol{\omega}^\top \mathbf{x}_i)\} + \frac{\beta}{2}\boldsymbol{\omega}^\top\boldsymbol{\omega}, \tag{14}$$

where the quadratic term in the model parameters corresponds to the prior. We must be careful when defining the precision term β, as the influence of the prior on the sampling can be significant as we show in Appendix D.1.1. Also, we allow the data pairs $\{y, \mathbf{x}\}_{i=1}^N$ to be augmented such that $x_0 = 1$, which allows ω_0 to correspond to the bias term. We then generate the data by separately sampling $\hat{\boldsymbol{\omega}} \sim \mathcal{N}(\mathbf{m}_{\boldsymbol{\omega}}, \boldsymbol{\Sigma}_{\boldsymbol{\omega}})$ and building a data set through using the logistic $y_i = 1/\left(1 + \exp(-\hat{\boldsymbol{\omega}}^\top \mathbf{x}_i)\right)$.

6.1.1 1D example

In a simple example, where $\boldsymbol{\omega} \in \mathbb{R}^2$ consists of a weight w and a bias term b, we show how RMHMC is able to take advantage of the geometry of the parameter space, whereas HMC does not. We see this in Figure 1, where we purposely initialise all samplers with $w = 10$ and $b = 0$ in order to show how each implementation manages when heading towards the known means of $w = 2$ and $b = 0$. There is a clear difference between the standard HMC sampler and both Riemannian samplers. The incorporation of geometry in both RMHMC schemes avoids the slower route that the HMC scheme follows. Furthermore HMC is renowned for being sensitive to the hyperparameters, such as the step size and the number of leapfrog steps. Therefore despite the fact that RMHMC still requires optimisation of these hyperparameters, it is less sensitive since the step size is automatically adapted according to the curvature of the space.

However, despite the obvious advantages of RMHMC, it comes at a severe computational cost. On the same hardware, the samples in Figure 1 take 0.5 s, 15.3 s and 9.3 s for HMC, Implict RMHMC and Explicit RMHMC respectively.[5] This is when allowing the fixed-point routine to take a maximum number of six iterations. Therefore although RMHMC is expensive, even this small example shows that Explicit RMHMC can help in reducing the computational cost.

[5]CPU: Intel i7-8700k; GPU: NVIDIA TITAN Xp; RAM: 32 GB

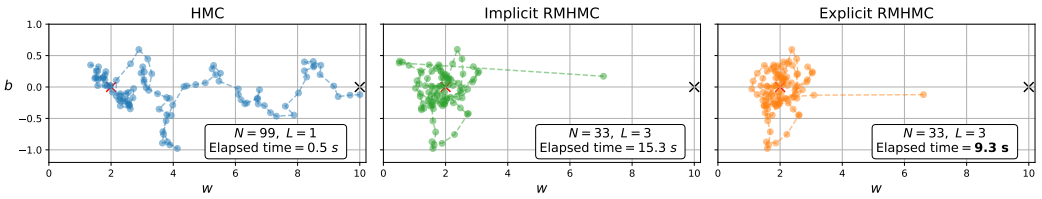


Figure 1: Comparison across the three Hamiltonian-based integration schemes, where all inference engines provide the same number of samples and are initialised with the same weight. This is a logistic regression example, where the weight space is two dimensional and samples are drawn from $\mathcal{N}([2, 0], 0.3\mathbf{I})$. The red cross marks the known mean from the generative model. This figure shows how the geometric knowledge of the space in both Riemannian models accelerates the particle to the true distribution, whereas the standard HMC model takes a more circuitous route. We highlight the elapsed time, which show a the advantage of Explicit RMHMC.

6.1.2 Influence of the prior in a 10D example

Although well known in Bayesian modelling, we use a 10D example to highlight the importance of the prior when designing the sampling scheme in HMC. We find that large values of β focus the samples near the zero-mean prior, whereas for a smaller value of β, we see the samples moving towards $\mathbf{m}_{\boldsymbol{\omega}}$. We display these results in Figure 6 in Appendix D.1.1 in order to highlight how the prior influences results. In addition, when setting the hyperparameters, we initialise all schemes with a long trajectory length and use the autocorrelation to set L. We aim to reduce the autocorrelation to result in a larger effective sample size. Appendix D.1 provides further details.

6.1.3 The significance of Ω

We cannot introduce a new parameter Ω without giving some intuition and empirical example of its influence in practice. We are able to show that as we vary the value of Ω, small values below a certain threshold display poor long-term performance, whereas above a threshold, we see a long-term performance consistent with the implicit integration scheme. We also see that when Ω is set too large, the long-term performance once again degrades. These results are consistent with Pihajoki [31] and Tao [32], and we refer to Figure 7 in Appendix D.1.2.

6.2 Inference in a Bayesian hierarchical model

We now introduce the funnel distribution, which is defined as

$$\prod_i \mathcal{N}(\mathbf{x}_i|0, \exp\{-v\})\mathcal{N}(v|0, 9).$$

This was first introduced by Neal et al. [35] and is a good example of a model that is challenging for Bayesian inference. Additionally, the marginal distribution of v is $\mathcal{N}(0, 9)$, thus allowing us to make comparisons to the known distribution. To demonstrate the advantages of RMHMC over HMC, we use the KL-divergence, $\mathrm{D}_{\mathrm{KL}}(p(v)\|q(v))$, as our metric of performance, as well as comparing the wall-clock time. To define $q(v)$, we take the first and second moments of the samples to define our approximation to the known normal distribution $p(v)$. The KL-divergence is appropriate because in a real application we would only have access to q, and it is in our interest to know how much information is lost if we rely on q instead of p.

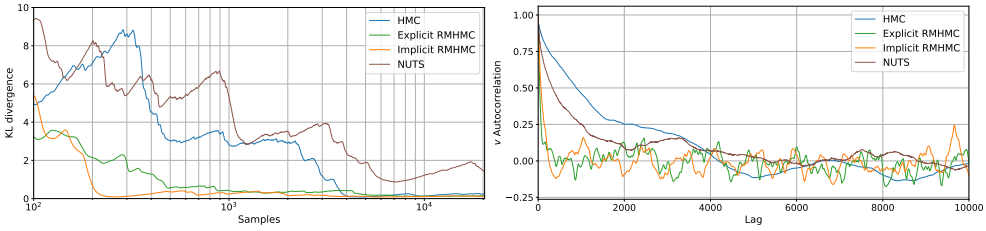
We compare four Hamiltonian-based Monte Carlo schemes, where our baseline is the HMC implementation of the No-U-Turn Sampler [18], which has been set to adapt its acceptance rate to between $0.65 - 0.9$.[6] We further provide an implementation of HMC that has been tuned with knowledge of the true marginal. Importantly, we compare *implicit RMHMC* with our new *explicit RMHMC* sampler. We further highlight that for both Riemannian schemes the Fisher information matrix is no longer

[6]The same parameter settings are followed as in Hoffman and Gelman [18]. The desired acceptance rate is set to 0.75, but this results in large step sizes that cause trajectories to traverse into numerically unstable areas of the log probability space. Therefore the adapted acceptance rate is higher than would normally be desired, although results in the same poor performance.

Table 1: $10 + 1$ dimensional funnel: comparison across integration schemes for inference in the funnel distribution. RMHMC outperforms HMC, even when HMC is optimised with a posteriori information. Furthermore explicit RMHMC achieves comparable performance to implicit RMHMC in almost a third of the time.

Scheme	D_{KL}	Wall Time	Acc. Rate	N	L	ϵ	α
HMC - NUTS	1.109	42 min	0.87	10^5	25	0.3896	-
Implicit RMHMC	0.130	3 hr 10 min	0.93	10^3	25	0.1500	10^6
Explicit RMHMC, $\Omega = 10$	0.142	1 hr 12 min	0.81	10^3	25	0.1400	10^6
HMC (Hand-Tuned for KL)	0.270	38 min	0.97	10^5	25	0.2	-

guaranteed to be positive semi-definite as it was for Bayesian logistic regression. Therefore we filter out negative eigenvalues by implementing *SoftAbs*, as in Betancourt [27], the details of which are available in Appendix C.



(a) KL divergence versus samples.

(b) Autocorrelation of v versus the lag.

Figure 2: Funnel experiment sampler diagnostics.

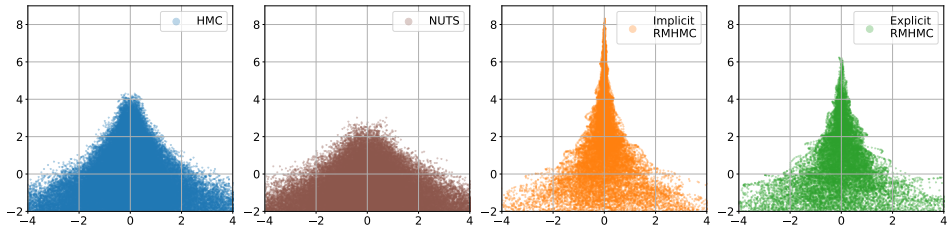


Figure 3: Comparison across Hamiltonian-based schemes for inference in a Bayesian hierarchical model. The Riemannian-based integrators have knowledge of the model's complex geometry and therefore more efficiently explore the space and are able to sample in the narrow part of the funnel. Our explicit integration scheme gives a significant speed-up over the previously used implicit scheme for the same number of samples. The No-U-Turn Sampler (NUTS) adapts its step size to a value that prevents it from traversing up the neck of the funnel.

Table 1 shows that both forms of RMHMC outperform HMC in terms of D_{KL}, despite providing the optimiser for HMC with a posteriori information by hand-tuning with knowledge of the true distribution and by allowing it to take two orders of magnitude more samples. The No-U-Turn Sampler, which adapts for an appropriate acceptance rate leads to too large a step size and prevents it from exploring the narrow neck of the funnel (Figure 3). However, the key result is in the wall-clock time, where our new explicit RMHMC achieves comparable $\mathrm{D}_{\mathrm{KL}}(p(v)\|q(v))$ to implicit RMHMC in almost a third of the time.[7] We further show both the convergence rate of each sampler in $\mathrm{D}_{\mathrm{KL}}(p(v)\|q(v))$ and the autocorrelations in Figure 2, where both RMHMC samplers display comparable performance in convergence and autocorrelation. The faster convergence of

[7]When the max number of fixed point iterations for Implicit RMHMC was set to 6 the acceptance rate for the same initial conditions dropped from 0.93 to 0.50, therefore the max number was set to 1000, with the convergence threshold for breaking out of the loop set to 0.001. At an acceptance rate of 0.50, the resulting sampler did achieve a similar performance in terms of wall time to Explicit RMHMC but this is clearly not a desirable rate.

$\mathrm{D}_{\mathrm{KL}}(p(v)\|q(v))$ for both RMHMC methods demonstrates how the samplers are more efficient at building an empirical representation of $p(v)$, when compared to the other HMC samplers. This is also highlighted by how the autocorrelation between samples plateaus around zero faster for Explicit and Implicit RMHMC. This suggests that the effective sample size of both these samplers is higher than for NUTS and hand-tuned HMC. Finally, the estimated marginal distributions of $p(v)$ are shown in Figure 4, where the better performance of the two Riemannian samplers can be seen from the lower KL divergence.

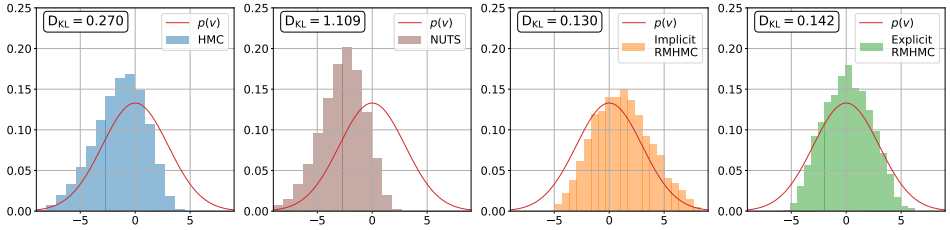


Figure 4: The estimated marginal distributions of $p(v)$ for all four samplers. Compared to the known distribution (the red line), the Riemannian samplers provide samples that appear less biased by the narrowness of the funnel. This leads to a lower (better) KL divergence.

7 Conclusion

By introducing an explicit symplectic integration scheme, we have reduced the number of higher-order derivative calculations required per leapfrog step for RMHMC. This results in an almost two times speed up over the previously used implicit integration scheme for RMHMC, with comparable performance. We have demonstrated these results both for Bayesian logistic regression and a Bayesian hierarchical model, where we have also provided insights as to how one might use explicit RMHMC in practice.

Finally, by reducing the computational burden of RMHMC, we hope to expand the range of models that may currently be too computationally expensive for RMHMC and enable others to do the same with our open-source Python package `hamiltorch`. We will explore these models in future work, with the objective of providing the tools for improving inference and achieving better uncertainty estimates.

Hamiltorch: a PyTorch Python package for sampling. All the Monte Carlo methods implemented in this paper were performed using `hamiltorch`. This is a Python package we developed that uses Hamiltonian Monte Carlo (HMC) to sample from probability distributions. `hamiltorch` is built with a PyTorch [6] backend to take advantage of the innate automatic differentiation. PyTorch is a machine learning framework that makes modelling with neural networks in Python easier. Therefore since `hamiltorch` is based on PyTorch, we ensured that `hamiltorch` was built to sample directly from neural network models (i.e. objects inheriting from the `torch.nn.Module`).

To use `hamiltorch` to sample a Bayesian neural network one only needs to define a data set $\{\mathbf{X}, \mathbf{Y}\}$ and a neural network model. PyTorch allows us to run much of the code on a GPU, which results in a large speed increase over running on a CPU. For example, if we use the Pytorch MNIST CNN example [36] based on the LeNet architecture [37], for the GPU we are able to achieve 116.33 samples per second, whereas on the CPU the sampling rate is 13.92 samples per second. This is for a network that consists of two convolutional layers and two fully connected layers, which contains $431{,}080$ weights (including biases).

Finally, it is simple to switch between integration schemes by changing the argument passed to the sampler. Therefore easily switching between HMC and RMHMC is part of our framework. The structure of `hamiltorch` aims to make performing HMC more scalable and accessible to practitioners, as well as enabling researchers to add their own metrics for RMHMC. For the code base please refer to `https://github.com/AdamCobb/hamiltorch` and for a tutorial please refer to `https://adamcobb.github.io/journal/hamiltorch.html`.

References

[1] Christopher KI Williams and Carl Edward Rasmussen. *Gaussian Processes for Machine Learning*, volume 2. MIT Press Cambridge, MA, 2006.

[2] Matthew D Hoffman, David M Blei, Chong Wang, and John Paisley. Stochastic variational inference. *The Journal of Machine Learning Research*, 14(1):1303–1347, 2013.

[3] David JC MacKay. A practical Bayesian framework for backpropagation networks. *Neural computation*, 4 (3):448–472, 1992.

[4] Mark Girolami and Ben Calderhead. Riemann manifold Langevin and Hamiltonian Monte Carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(2):123–214, 2011.

[5] Radford M Neal et al. MCMC using Hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*, 2 (11):2, 2011.

[6] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. 2017.

[7] Yarin Gal and Lewis Smith. Sufficient conditions for idealised models to have no adversarial examples: a theoretical and empirical study with Bayesian neural networks. *arXiv preprint arXiv:1806.00667*, 2018.

[8] Cheng Li, Sanvesh Srivastava, and David B Dunson. Simple, scalable and accurate posterior interval estimation. *Biometrika*, 104(3):665–680, 2017.

[9] David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877, 2017.

[10] Adam D Cobb, Stephen J Roberts, and Yarin Gal. Loss-calibrated approximate inference in Bayesian neural networks. *arXiv preprint arXiv:1805.03901*, 2018.

[11] Nicholas Metropolis, Arianna W Rosenbluth, Marshall N Rosenbluth, Augusta H Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21(6): 1087–1092, 1953.

[12] W Keith Hastings. Monte Carlo sampling methods using Markov chains and their applications. 1970.

[13] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006. ISBN 0387310738.

[14] Simon Duane, Anthony D Kennedy, Brian J Pendleton, and Duncan Roweth. Hybrid Monte Carlo. *Physics letters B*, 195(2):216–222, 1987.

[15] Berni J Alder and Thomas Everett Wainwright. Studies in molecular dynamics. I. General method. *The Journal of Chemical Physics*, 31(2):459–466, 1959.

[16] Radford M Neal. *BAYESIAN LEARNING FOR NEURAL NETWORKS*. PhD thesis, University of Toronto, 1995.

[17] Gareth O Roberts and Osnat Stramer. Langevin diffusions and Metropolis-Hastings algorithms. *Methodology and computing in applied probability*, 4(4):337–357, 2002.

[18] Matthew D Hoffman and Andrew Gelman. The No-U-Turn sampler: adaptively setting path lengths in Hamiltonian Monte Carlo. *Journal of Machine Learning Research*, 15(1):1593–1623, 2014.

[19] Michael J Betancourt. Generalizing the no-U-turn sampler to Riemannian manifolds. *arXiv preprint arXiv:1304.1920*, 2013.

[20] Ziyu Wang, Shakir Mohamed, and Nando Freitas. Adaptive Hamiltonian and Riemann Manifold Monte Carlo. In *International Conference on Machine Learning*, pages 1462–1470, 2013.

[21] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical Bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, pages 2951–2959, 2012.

[22] Babak Shahbaba, Shiwei Lan, Wesley O Johnson, and Radford M Neal. Split Hamiltonian Monte Carlo. *Statistics and Computing*, 24(3):339–349, 2014.

[23] Shiwei Lan, Vassilios Stathopoulos, Babak Shahbaba, and Mark Girolami. Lagrangian Dynamical Monte Carlo. *arXiv preprint arXiv:1211.3759*, 2012.

[24] Ernst Hairer, Christian Lubich, and Gerhard Wanner. *Geometric numerical integration: structure-preserving algorithms for ordinary differential equations*, volume 31. Springer Science & Business Media, 2006.

[25] Shun-Ichi Amari and Hiroshi Nagaoka. Methods of information geometry. 2000.

[26] Benedict Leimkuhler and Sebastian Reich. *Simulating Hamiltonian dynamics*, volume 14. Cambridge university press, 2004.

[27] Michael Betancourt. A general metric for Riemannian manifold Hamiltonian Monte Carlo. In *International Conference on Geometric Science of Information*, pages 327–334. Springer, 2013.

[28] Molei Tao. Explicit high-order symplectic integrators for charged particles in general electromagnetic fields. *Journal of Computational Physics*, 327:245–251, 2016.

[29] Junjie Luo, Xin Wu, Guoqing Huang, and Fuyao Liu. Explicit Symplectic-like Integrators with Midpoint Permutations for Spinning Compact Binaries. *The Astrophysical Journal*, 834(1):64, 2017.

[30] Ruili Zhang, Yulei Wang, Yang He, Jianyuan Xiao, Jian Liu, Hong Qin, and Yifa Tang. Explicit symplectic algorithms based on generating functions for relativistic charged particle dynamics in time-dependent electromagnetic field. *Physics of Plasmas*, 25(2):022117, 2018.

[31] Pauli Pihajoki. Explicit methods in extended phase space for inseparable Hamiltonian problems. *Celestial Mechanics and Dynamical Astronomy*, 121(3):211–231, 2015.

[32] Molei Tao. Explicit symplectic approximation of nonseparable Hamiltonians: Algorithm and long time performance. *Physical Review E*, 94(4):043303, 2016.

[33] Gilbert Strang. On the construction and comparison of difference schemes. *SIAM Journal on Numerical Analysis*, 5(3):506–517, 1968.

[34] Haruo Yoshida. Construction of higher order symplectic integrators. *Physics letters A*, 150(5-7):262–268, 1990.

[35] Radford M Neal et al. Slice sampling. *The annals of statistics*, 31(3):705–767, 2003.

[36] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. PyTorch MNIST example. `https://github.com/pytorch/examples/blob/master/mnist/main.py`, 2017. Accessed: 8th September 2019.

[37] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[38] Paul J Channell and Clint Scovel. Symplectic integration of Hamiltonian systems. *Nonlinearity*, 3(2):231, 1990.

[39] H Poincaré. Les Méthodes nouvelles de la Mécanique Céleste. *Paris, Grauthier-Viltars*, 3:46, 1899.

Introducing an Explicit Symplectic Integration Scheme for Riemannian Manifold Hamiltonian Monte Carlo (Supplementary Material)

A Derivation of distance metric

A common metric for measuring a distance between distributions is the Kullback–Leibler (KL)divergence,

$$\mathrm{D}_{\mathrm{KL}}\left(p(\mathbf{x},\boldsymbol{\omega})||p(\mathbf{x},\boldsymbol{\omega}')\right) = \iint p(\mathbf{x},\boldsymbol{\omega})\left(\log p(\mathbf{x},\boldsymbol{\omega}) - \log p(\mathbf{x},\boldsymbol{\omega}')\right)\mathrm{d}\mathbf{x}\mathrm{d}\boldsymbol{\omega}. \tag{15}$$

If we expand the $\log p(\mathbf{x},\boldsymbol{\omega}')$ term within the integrand using a Taylor expansion around $\boldsymbol{\omega}$ such that $\delta\boldsymbol{\omega} = \boldsymbol{\omega}' - \boldsymbol{\omega}$, we obtain

$$\begin{aligned}\mathrm{D}_{\mathrm{KL}}\left(p(\mathbf{x},\boldsymbol{\omega})||p(\mathbf{x},\boldsymbol{\omega}+\delta\boldsymbol{\omega})\right) \approx &\iint p(\mathbf{x},\boldsymbol{\omega})\log p(\mathbf{x},\boldsymbol{\omega})\mathrm{d}\mathbf{x}\mathrm{d}\boldsymbol{\omega} \\ &- \iint p(\mathbf{x},\boldsymbol{\omega})\Big(\log p(\mathbf{x},\boldsymbol{\omega}) + \left(\frac{\nabla_{\boldsymbol{\omega}} p(\mathbf{x},\boldsymbol{\omega})}{p(\mathbf{x},\boldsymbol{\omega})}\right)^{\top}\delta\boldsymbol{\omega} \\ &+ \frac{1}{2}\delta\boldsymbol{\omega}^{\top}\nabla^2_{\boldsymbol{\omega}}\log p(\mathbf{x},\boldsymbol{\omega})\delta\boldsymbol{\omega} + \dots\Big)\mathrm{d}\mathbf{x}\mathrm{d}\boldsymbol{\omega}.\end{aligned} \tag{16}$$

We note that the first two terms in Equation (16) cancel and the first order gradient term reduces to zero as follows,

$$\begin{aligned}-\iint p(\mathbf{x},\boldsymbol{\omega})\left(\frac{\nabla_{\boldsymbol{\omega}} p(\mathbf{x},\boldsymbol{\omega})}{p(\mathbf{x},\boldsymbol{\omega})}\right)^{\top}\delta\boldsymbol{\omega}\mathrm{d}\mathbf{x}\mathrm{d}\boldsymbol{\omega} &= -\iint \nabla_{\boldsymbol{\omega}} p(\mathbf{x},\boldsymbol{\omega})^{\top}\delta\boldsymbol{\omega}\mathrm{d}\mathbf{x}\mathrm{d}\boldsymbol{\omega} = -\int \nabla_{\boldsymbol{\omega}} p(\boldsymbol{\omega})^{\top}\delta\boldsymbol{\omega}\mathrm{d}\boldsymbol{\omega} \\ &= -\nabla_{\boldsymbol{\omega}}\mathbf{1}^{\top}\delta\boldsymbol{\omega} \\ &= 0.\end{aligned} \tag{17}$$

Thus,

$$\begin{aligned}\mathrm{D}_{\mathrm{KL}}\left(p(\mathbf{x},\boldsymbol{\omega})||p(\mathbf{x},\boldsymbol{\omega}+\delta\boldsymbol{\omega})\right) &\approx -\iint p(\mathbf{x},\boldsymbol{\omega})\left(\frac{1}{2}\delta\boldsymbol{\omega}^{\top}\nabla^2_{\boldsymbol{\omega}}\log p(\mathbf{x},\boldsymbol{\omega})\delta\boldsymbol{\omega}\right)\mathrm{d}\mathbf{x}\mathrm{d}\boldsymbol{\omega} \\ &= -\iint p(\mathbf{x},\boldsymbol{\omega})\left(\frac{1}{2}\delta\boldsymbol{\omega}^{\top}\mathbf{G}(\mathbf{x},\boldsymbol{\omega})\delta\boldsymbol{\omega}\right)\mathrm{d}\mathbf{x}\mathrm{d}\boldsymbol{\omega},\end{aligned} \tag{18}$$

which results in the outcome of Equation (8).

B Checking the symplectomorphism of the augmented Hamiltonian binding term

In order for a transformation matrix to be symplectic, it must satisfy the condition Channell and Scovel [38]:

$$\mathbf{H}^{\top}\mathbf{J}\mathbf{H} = \mathbf{J} \tag{19}$$

where,

$$\mathbf{J} = \begin{bmatrix}\mathbf{0} & \mathbf{I} \\ -\mathbf{I} & \mathbf{0}\end{bmatrix} \tag{20}$$

is a non-singular, skew-symmetric matrix and $\mathbf{H} \in \mathbb{R}^{2D\times 2D}$ is a symplectic matrix. This definition [24, Chapter 6, Page 183] is derived from the area preserving properties of the linear mapping $\mathbf{H}$, where the structure of $\mathbf{J}$ originates from the form of the determinant in the transformed space. In nonlinear systems, $\mathbf{H}$ can be the Jacobian of a differentiable function [39] as shown in [24, Chapter 6, Page 184].

To show how the binding term in Equation (10) meets this condition, we can transform the Hamiltonian $\tilde{H}(\boldsymbol{\omega},\mathbf{p},\tilde{\boldsymbol{\omega}},\tilde{\mathbf{p}})$ to $\hat{H}(\hat{\boldsymbol{Q}},\hat{\boldsymbol{P}},\boldsymbol{\alpha},\boldsymbol{\beta})$, where $\hat{\boldsymbol{Q}} = \boldsymbol{\omega} + \tilde{\boldsymbol{\omega}}$, $\hat{\boldsymbol{P}} = \mathbf{p} + \tilde{\mathbf{p}}$, $\boldsymbol{\alpha} = \boldsymbol{\omega} - \tilde{\boldsymbol{\omega}}$ and $\boldsymbol{\beta} = \mathbf{p} - \tilde{\mathbf{p}}$, giving

$$\hat{H}(\hat{\boldsymbol{Q}},\hat{\boldsymbol{P}},\boldsymbol{\alpha},\boldsymbol{\beta}) = H_1(\hat{\boldsymbol{Q}} + \boldsymbol{\alpha}/2, \hat{\boldsymbol{P}} - \boldsymbol{\alpha}/2) + H_2(\hat{\boldsymbol{Q}} - \boldsymbol{\alpha}/2, \hat{\boldsymbol{P}} + \boldsymbol{\alpha}/2) + \Omega\frac{1}{2}(\|\boldsymbol{\alpha}\|^2 + \|\boldsymbol{\beta}\|^2) + \mathcal{R} \tag{21}$$

as shown in [32], where $\mathcal{R}$ is the perturbative higher order remainder.

Therefore, if we employ this transformation of variables to the binding term $\phi^{\delta}_{\Omega h}$, the linear system in Equation (13) corresponding to $\phi^{\delta}_{\Omega h}$ becomes:

$$\begin{bmatrix}\hat{\boldsymbol{Q}} \\ \hat{\boldsymbol{P}} \\ \boldsymbol{\alpha} \\ \boldsymbol{\beta}\end{bmatrix} = \begin{bmatrix}1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \cos(2\Omega\delta) & \sin(2\Omega\delta) \\ 0 & 0 & -\sin(2\Omega\delta) & \cos(2\Omega\delta)\end{bmatrix}\begin{bmatrix}\hat{\boldsymbol{Q}} \\ \hat{\boldsymbol{P}} \\ \boldsymbol{\alpha} \\ \boldsymbol{\beta}\end{bmatrix} \tag{22}$$

after the manipulation of the variables from the linear system in (12).

We will now write this as

$$\mathbf{y}' = \mathbf{H}\mathbf{y}, \tag{23}$$

where $\mathbf{H}$ is the transformation matrix in Equation (22). In order to check this transformation satisfies the condition in Equation (20), we do $\mathbf{H}^\top \mathbf{J}\mathbf{H}$. After applying these matrix multiplications and using the trigonometric identity $\sin^2(x) + \cos^2(x) = 1$ we see that $\mathbf{H}^\top \mathbf{J}\mathbf{H} = \mathbf{J}$ thus confirming $\phi^\delta_{\Omega h}$ is a symplectomorphism.

C Hessians with negative eigenvalues

In Bayesian logistic regression, we can calculate the Fisher information matrix with knowledge that the metric will be positive semi-definite Girolami and Calderhead [4]. However, once models become more complex, the positive semi-definiteness of the Fisher information metric is no longer guaranteed.

In order to ensure $\mathbf{G}(\boldsymbol{\omega})$ is positive semi-definite, we follow the the same *SoftAbs* metric that [27] introduced for RMHMC. This allows us to filter the eigenspectrum by passing all the eigenvalues, λ_d, through the function

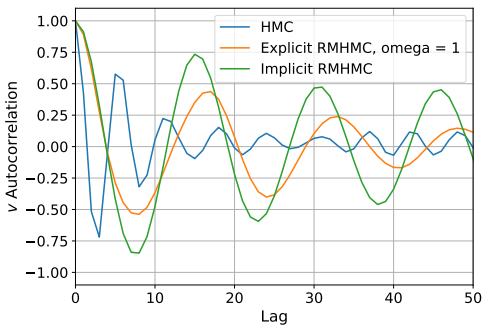
$$\tilde{\lambda}_d = \lambda_d \coth(\alpha \lambda_d). \tag{24}$$

This approximates the absolute values of the eigenvalues and therefore ensures that the new metric $\tilde{\mathbf{G}}(\boldsymbol{\omega}) = \mathbf{Q}\tilde{\boldsymbol{\lambda}}\mathbf{Q}^\top$ is positive semi-definite, by introducing a parameter α which controls the smoothness. As $\alpha \to \infty$ the SoftAbs function becomes the absolute value. In our work we set $\alpha = 10^6$ as in [27].

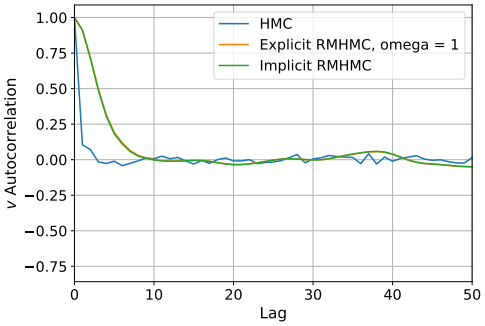
D Experiments

D.1 Bayesian logistic regression

To tune the trajectory length, L we started with $L = 30$ and ran for 100 iterations. We then looked at the autocorrelation in Figure 5a and set L to less than half the period of the oscillations for each respective inference scheme (HMC: $L = 1, \epsilon = 0.2$. Implicit RMHMC: $L = 3, \epsilon = 0.04$. Explicit RMHMC: $L = 3, \epsilon = 0.04$.). This led us to Figure 5b, where the autocorrelation is significantly reduced.



(a) $L = 30$ and results in oscillatory behaviour.



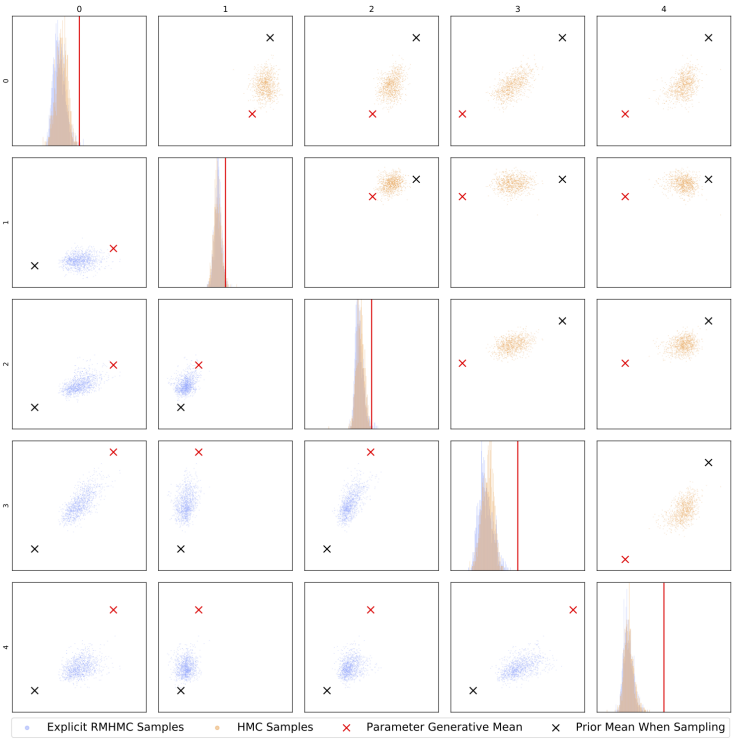
(b) L is reduced to less than half the period for each respective scheme.

Figure 5: Autocorrelation for Bayesian logistic regression, where $\boldsymbol{\omega} \in \mathbb{R}^{11}$.

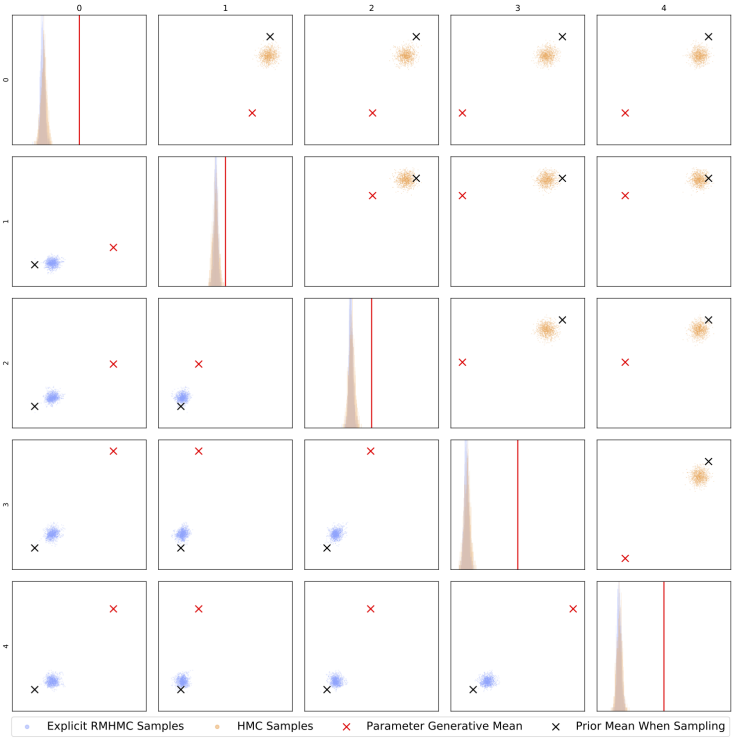
D.1.1 Dependence on the prior and the initialisation of RMHMC

In this logistic regression example the assumed prior over the combined parameter vector, $\boldsymbol{\omega}$, is given by $\mathcal{N}(\mathbf{0}, \beta\mathbf{I})$. We note that this is compared to the known generative model $\hat{\boldsymbol{\omega}} \sim \mathcal{N}(\mathbf{m}_\omega, \boldsymbol{\Sigma}_\omega)$ when we build the data set. The value of β defines the confidence in the prior. We expect that for large values of β, the structure of the prior will dominate the when inferring the parameters, whereas small values of β are expected to result in broader, less-informative priors that reduce in their influence in the presence of more data. We display this behaviour in Figure 6, where we display the samples for five of the eleven parameters.

Furthermore, it often benefits the RMHMC schemes to be initialised by a cheap HMC burn-in to ensure that the linear algebra associated with the Hessian is numerically stable (we found that far away initialisations caused numerical issues due to the Hessian becoming singular).



(a) Weak Prior, $\beta = 0.2$.



(b) Strong Prior, $\beta = 10$.

Figure 6: A comparison between Explicit RMHMC and HMC for Bayesian logistic regression, where $\boldsymbol{\omega} \in \mathbb{R}^{11}$. The purpose of these plots is to both demonstrate how HMC and RMHMC compare in higher dimensions and to show how the prior affects the sampling. We note that we display the results of Explicit RMHMC rather than Implicit RMHMC due to their similar performance. The HMC samples tend to be more spread than the RMHMC samples, however the RMHMC samples lead to slightly better accuracy performance on the logistic regression task.

D.1.2 Implications of the binding term on performance

For the single 1D case, such that $w \in \mathbb{R}^1$ and $b \in \mathbb{R}^1$, we can test how the performance of the binding term of the explicit integrator (Ω) affects long term performance. If we initialise the samplers at the location $[0, 0]$ in parameter space and set the trajectory length to 40, we can observe how different values of Ω affect the trajectory. In Figure 7, we compare the implicit integration scheme in Equation (6) to various explicit schemes with different values for Ω. We treat the dashed red line, Implicit RMHMC, as the ground truth and show how all other explicit RMHMC trajectories compare. It is clear from this simple example, that Ω has an important implication on the long term performance of RMHMC. This was also pointed out in the original paper by Tao [32], whereby Ω must be larger than some threshold Ω_0 but also small enough such that the error bound $\mathcal{O}(N\delta^2\Omega)$ remains small. Figure 7 shows what this means in practice.

In fact the purpose of the additional binding term was to improve on the long term performance of the integration scheme introduced by [31], which is equivalent to $\Omega = 0$. We can clearly see this is the case, as the trajectory corresponding to $\Omega = 0$ in Figure 7 diverges from the implicit scheme shortly after traversing away from the initial condition. Furthermore, the largest value we were able to set for the binding term before the trajectory was rejected, $\Omega = 472$, also diverges from the implicit scheme.

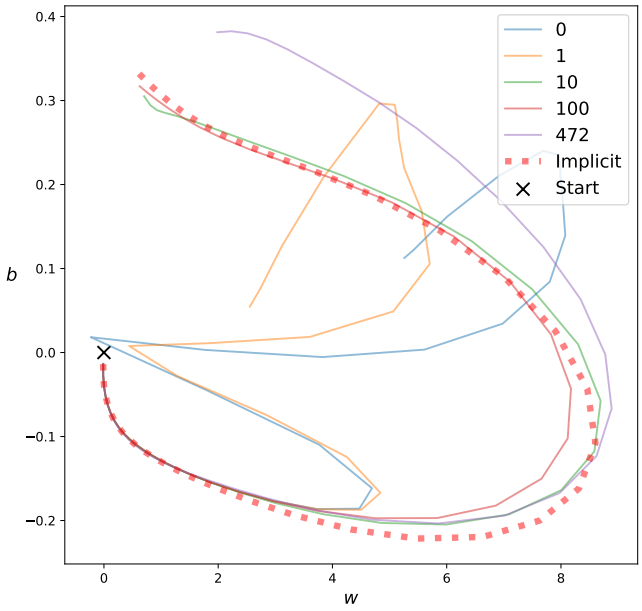


Figure 7: Long-term performance of explicit RMHMC when comparing to Implicit RMHMC for a single trajectory ($L = 40, \epsilon = 0.012$). Small values of Ω, as indicated in the legend, diverge shortly after the initial conditions, as well as the largest value.