Qualifying Paper 2

Metropolis Adjusted Microcanonical Hamiltonian Monte Carlo

Professor: Trevor Campbell

Isaac Rankin

December 4, 2025

Contents

1 Introduction

Suppose we wish to draw samples from a possibly unnormalised density $p : \mathbb{R}^d \to \mathbb{R}$. Markov chain Monte Carlo (MCMC) algorithms construct a sequence $\{\boldsymbol{x}_i\}_{i=1}^n$ where each $\boldsymbol{x}_i \in \mathbb{R}^d$ [1]. When the MCMC algorithm is designed carefully, the empirical distribution of the sequence, or chain, approximates p. In many applications, practitioners desire [2, 3] that asymptotically, the distribution of chain converges to p, ensuring that the samples can be used to make unbiased inferences.

When gradient information is available, gradient-based MCMC methods can achieve substantially better efficiency than random-walk methods, particularly in high dimensions [1, 4]. For interdisciplinary applications, automatic hyperparameter tuning schemes are highly desirable, as they allow practitioners to obtain samples without requiring expertise in the algorithm's implementation. Such automatically tuned algorithms are often referred to as *blackbox*.

Robnik, Cohn-Gordon, and Seljak (RCS) propose the Metropolis-adjusted microcanonical sampler (MAMS) [3], which adds a Metropolis-Hastings correction to the earlier asymptotically biased microcanonical Hamiltonian Monte Carlo (MCHMC) and its variant, microcanonical Langevin-like Hamiltonian Monte Carlo (MCLMC). The primary contribution is deriving the correct acceptance probability for dynamics whose numerical integrator is not volume-preserving.

This report critically examines the manuscript *Metropolis Adjusted Microcanonical Hamiltonian Monte Carlo* [3] which introduced MAMS. We analyze the technical details, discuss the contributions of the work, and identify limitations and unanswered questions regarding the novel algorithm. We supplement this analysis with a project comparing RCS's hyperparameter tuning scheme to a Bayesian optimization approach.

2 Hamiltonian Monte Carlo and related work

We shall now briefly introduce Hamiltonian Monte Carlo (HMC) and discuss work related to that of RCS.

We write the target density as $p(\boldsymbol{x}) = e^{-\mathcal{L}(\boldsymbol{x})}/Z$ where $Z = \int e^{-\mathcal{L}(\boldsymbol{x})} d\boldsymbol{x}$ is the normalizing constant, and $\mathcal{L}(\boldsymbol{x}) = -\log p(\boldsymbol{x}) - \log Z$. Note that gradients of p are proportional to those of $\mathcal{L}$: $\nabla\mathcal{L}(\boldsymbol{x}) = -\nabla p(\boldsymbol{x})/p(\boldsymbol{x}) \propto \nabla p(\boldsymbol{x})$.

HMC augments the target 'position' $\boldsymbol{x}$ with an ancillary 'momentum' $\boldsymbol{u} \in \mathbb{R}^d$, forming a so-called 'phase-space' [1, 4]. Following RCS's nomenclature, we refer to the standard HMC dynamics as the 'canonical' Hamiltonian dynamics: $\dot{\boldsymbol{x}} = \boldsymbol{u}$, $\dot{\boldsymbol{u}} = -\nabla\mathcal{L}(\boldsymbol{x})$, where $\dot{\boldsymbol{x}}$ and $\dot{\boldsymbol{u}}$ are the time derivatives of the position and momentum respectively. A solution to this system of differential equations corresponds to trajectories on level sets of the separable Hamiltonian $H(\boldsymbol{x}, \boldsymbol{u}) = \mathcal{L}(\boldsymbol{x}) + \frac{1}{2}\boldsymbol{u}^T\boldsymbol{u}$. Typically, we write $H(\boldsymbol{x}, \boldsymbol{u}) = V(\boldsymbol{x}) + K(\boldsymbol{u})$, where the potential energy is $V(\boldsymbol{x}) = \mathcal{L}(\boldsymbol{x})$, and the kinetic energy is $K(\boldsymbol{u}) = \frac{1}{2}\boldsymbol{u}^T\boldsymbol{u}$ [1]. The stationary distribution describing $(\boldsymbol{x}, \boldsymbol{u})$ on phase-space is $p(\boldsymbol{x})\,\mathcal{N}(\boldsymbol{u}; \boldsymbol{0}, \boldsymbol{I})$ [1, 4]. Crucially, the numerical integrator for canonical HMC, typically the leapfrog (Verlet) integrator, is symplectic [1, 3, 4], that is, it preserves phase-space volume and so its corresponding Jacobian determinant is one, simplifying the Metropolis-Hastings acceptance probability calculation.

Robnik and Seljak, collaborating with Bruno De Luca and Silverstein, previously introduced microcanonical Hamiltonian Monte Carlo (MCHMC) and microcanonical Langevin-like Hamiltonian Monte Carlo (MCLMC) [2], which simulate 'microcanonical dynamics' with unit-norm momentum $\|\boldsymbol{u}\| = 1$. The microcanonical dynamics are defined as:

$$\dot{\boldsymbol{x}} = \boldsymbol{u}, \qquad \dot{\boldsymbol{u}} = -\frac{I - \boldsymbol{u}\boldsymbol{u}^T}{d-1}\nabla\mathcal{L}(\boldsymbol{x}).$$

The projection operator $(I - \boldsymbol{u}\boldsymbol{u}^T)/(d-1)$ ensures $\dot{\boldsymbol{u}} \perp \boldsymbol{u}$, preserving $\|\boldsymbol{u}\| = 1$ and constraining the momentum to the unit sphere S^{d-1} [2, 3, 5, 6]. When integrated exactly, these dynamics have stationary distribution $p(\boldsymbol{x})\,\mathcal{U}_{S^{d-1}}(\boldsymbol{u})$ [2]. In contrast to canonical HMC, the leapfrog integrator for microcanonical dynamics is not

symplectic [2, 3, 4]. The projection operator compresses or expands the phase-space volume depending on alignment between momentum and gradient. This means the Jacobian determinant is not one, and computing the acceptance probability requires additional work. The prior work on MCHMC and MCLMC avoided this issue by not including a Metropolis-Hastings step, accepting the resulting asymptotic bias [2, 3, 5, 6]. Note that the algorithms MCHMC and MCLMC differ by momentum resampling. MCHMC samples a unit-norm momentum vector every $k \in \mathbb{N}$ trajectories whereas MCLMC pertubes the momentum frequently during the trajectory [2, 5]. Either MCHMC and MCLMC can be augmented with a Metropolis accept-reject step to ensure asymptotic unbiasedness [3]. From here on, we refer to MAMS as the Metropolis-adjusted MCHMC with $k = 1$.

For both the canonical and microcanonical dynamics, $p(\boldsymbol{x}) = \int p(\boldsymbol{x})\, \mathcal{N}(\boldsymbol{u}; \boldsymbol{0}, \boldsymbol{I}) d\boldsymbol{u} = \int p(\boldsymbol{x})\, \mathcal{U}_{S^{d-1}}(\boldsymbol{u}) d\boldsymbol{u}$ and so sampling from p is equivalent to simulating the dynamics to obtain a position-momentum pair $(\boldsymbol{x}, \boldsymbol{u})$ and discarding the momentum $\boldsymbol{u}$. As solving this system exactly is typically intractable, we use numerical integrators whose discretisation error requires a Metropolis-Hastings correction to ensure the sampler targets the joint distribution of $(\boldsymbol{x}, \boldsymbol{u})$ exactly [1, 4]. Note that RCS call on terminology from statistical physics, calling their method 'microcanonical' because the momentum magnitude is fixed, analogous to fixed energy in NVE (microcanonical) ensembles, rather than varying energy as for NVT (canonical) ensembles [2, 3, 7]

Several previous HMC variants have modified either the dynamics, or the mass structure (and hence the momentum). Note that by setting the 'mass' of the fictitious particle equal to one, we can use the terms 'momentum' and 'velocity' interchangeably. Riemannian manifold HMC [8] uses a position-dependent mass matrix $M(\boldsymbol{x})$ with kinetic energy $K(\boldsymbol{x}, \boldsymbol{u}) = \frac{1}{2}\boldsymbol{u}^T M(\boldsymbol{x})^{-1}\boldsymbol{u}$, adapting to local curvature. Relativistic HMC [9] uses $K(\boldsymbol{u}) = \sqrt{1 + \|\boldsymbol{u}\|^2}$ (with additional hyperparameters set to 1 here), bounding velocity for stability to large gradients. Magnetic HMC [10] introduces non-canonical dynamics, has an accept-reject step, but crucially the integrator for these dynamics is volume preserving. Magnetic HMC follows the dynamics $\dot{\boldsymbol{x}} = F\boldsymbol{u} + E\nabla U(\boldsymbol{x})$ $\dot{\boldsymbol{u}} = F^T \nabla U(\boldsymbol{x}) + G\boldsymbol{u}$ where $F, E,$ and$G$ form an antisymmetric block matrix, generalizing the canonical dynamics which can be written in matrix notation with the positive and negative identity matrix on the diagonal, and the zero matrix on the off diagonal. Pakman [11] explore non-standard dynamics, sampling momentum from the Laplace distribution yeilding a Hamiltonian (choosing constants equal to 1) $H(\boldsymbol{x}, \boldsymbol{u}) = -\cos(\boldsymbol{x}) + |\boldsymbol{u}|$. Ver Steeg and Galstyan chose a 'kinetic' energy $K(\boldsymbol{u}) = \frac{d}{2}\log\|\frac{\boldsymbol{u}}{d}\|$ [12]. They argued that in machine learning applications, enforcing physicallity is irrelevant [12], justifying their 'non-Newtonian' momentum. The microcanonical dynamics used by RCS are equivalent to those employed by Ver Steeg and Galstyan, though doubts have been raised as to whether their deterministic algorithm produces an ergodic chain [2, 3].

There have also been advancements in hyperparamater tuning schemes. Proposals are generated by numerically integrating the chosen dynamics, often using the leapfrog or Verlet integrator [1, 4]. The leapfrog integrator approximating the dynamics, requires tuning of step size ϵ and trajectory length L. In order to make an algorithm blackbox, ϵ and L, and in some cases a preconditioner matrix to make variability comparable across dimensions [13], must be selected without input from the practitioner. Too large of step sizes can cause instability and low acceptance rates; small step sizes waste computation through small moves causing slow exploration [14]. Trajectory lengths must balance exploration with efficiency to avoid retracing paths [15]. The No-U-Turn sampler (NUTS) [15] automates trajectory length selection by growing trajectories until they double back, eliminating manual tuning of L. Step size adaptation during warmup uses dual averaging [15, 16] based on acceptance rate tracking. There are many strategies for tuning hyperparameters. This problem will be the focus of the project in Section 7.

3 Deriving the acceptance probability

MAMS adds a Metropolis-Hastings acceptance step to MCHMC to eliminate asymptotic bias. The primary technical challenge–and main contribution of RCS–is deriving the correct acceptance probability for dynamics whose numerical integrator is not volume-preserving. For canonical HMC, the symplectic integrator ensures the Jacobian determinant equals one [1], simplifying the acceptance probability to a potential energy difference.

For microcanonical dynamics, this simplification does not hold.

The microcanonical dynamics are approximated using leapfrog integration with step size ϵ and trajectory length L, yielding $n = \lceil L/\epsilon \rceil$ steps. The leapfrog integrator is the composition:

$$\rho = \mathcal{T} \circ (B_{\epsilon/2} \circ A_\epsilon \circ B_{\epsilon/2}) \circ \cdots \circ (B_{\epsilon/2} \circ A_\epsilon \circ B_{\epsilon/2})$$

where

$$\begin{aligned} B_{\epsilon/2}: \quad & \boldsymbol{u} \leftarrow \boldsymbol{u} - \frac{\epsilon}{2} \frac{I - \boldsymbol{u}\boldsymbol{u}^T}{d-1} \nabla \mathcal{L}(\boldsymbol{x}), \\ A_\epsilon: \quad & \boldsymbol{x} \leftarrow \boldsymbol{x} + \epsilon \boldsymbol{u} \end{aligned}$$

and $(B_{\epsilon/2} \circ A_\epsilon \circ B_{\epsilon/2})$ is composed n times. After n steps, momentum is flipped: $\mathcal{T}(\boldsymbol{x}, \boldsymbol{u}) = (\boldsymbol{x}, -\boldsymbol{u})$. The integrator ρ is reversible, that is, an involution $\rho \circ \rho = id$, where id denotes the identity $id(\boldsymbol{x}) = \boldsymbol{x}$ for every $\boldsymbol{x} \in \mathbb{R}^d$. Crucially, unlike canonical HMC, the leapfrog integrator for microcanonical dynamics is *not volume-preserving*, necessitating computing a determinant of a Jacobian.

For a proposal $\boldsymbol{z}' = (\boldsymbol{x}', \boldsymbol{u}')$ from current state $\boldsymbol{z} = (\boldsymbol{x}, \boldsymbol{u})$, the required acceptance probability [1] is:

$$\min\left(1, \frac{p(\boldsymbol{z}')}{p(\boldsymbol{z})} \frac{q(\boldsymbol{z}|\boldsymbol{z}')}{q(\boldsymbol{z}'|\boldsymbol{z})}\right) = \min\left(1, e^{-W(\boldsymbol{z}', \boldsymbol{z})}\right).$$

Since $\boldsymbol{u}$ and $\boldsymbol{u}'$ are both uniformly distributed on S^{d-1}, we have $\mathcal{U}_{S^{d-1}}(\boldsymbol{u}) = \mathcal{U}_{S^{d-1}}(\boldsymbol{u}')$, and so the density ratio becomes:

$$\frac{p(\boldsymbol{z}')}{p(\boldsymbol{z})} = \frac{p(\boldsymbol{x}')\, \mathcal{U}_{S^{d-1}}(\boldsymbol{u}')}{p(\boldsymbol{x})\, \mathcal{U}_{S^{d-1}}(\boldsymbol{u})} = \frac{p(\boldsymbol{x}')}{p(\boldsymbol{x})} = e^{-(\mathcal{L}(\boldsymbol{x}') - \mathcal{L}(\boldsymbol{x}))}.$$

The proposal density ratio requires more careful treatment as ρ is not volumne preserving. As ρ is deterministic and reversible, $q(\boldsymbol{z}'|\boldsymbol{z}) = \delta(\boldsymbol{z}' - \rho(\boldsymbol{z}))$ and $q(\boldsymbol{z}|\boldsymbol{z}') = \delta(\boldsymbol{z} - \rho(\boldsymbol{z}'))$ where $\delta(\cdot)$ is the Dirac delta function. Using a change of variables:

$$\delta(\boldsymbol{z} - \rho(\boldsymbol{z}')) = \delta(\boldsymbol{z}' - \rho(\boldsymbol{z})) \left| \frac{\partial \rho}{\partial \boldsymbol{z}}(\boldsymbol{z}) \right|$$

therefore:

$$\frac{q(\boldsymbol{z}|\boldsymbol{z}')}{q(\boldsymbol{z}'|\boldsymbol{z})} = \frac{\delta(\boldsymbol{z} - \rho(\boldsymbol{z}'))}{\delta(\boldsymbol{z}' - \rho(\boldsymbol{z}))} = \left| \frac{\partial \rho}{\partial \boldsymbol{z}}(\boldsymbol{z}) \right|.$$

Applying Liouville's theorem to the microcanonical dynamics written as vector field $\dot{\boldsymbol{z}}(t) = F(\boldsymbol{z}(t)) = (\boldsymbol{u}(s), -\frac{I - \boldsymbol{u}(s)\boldsymbol{u}(s)^T}{d-1} \nabla \mathcal{L}(\boldsymbol{x}(s)))$ for which ρ approximates, he have:

$$\log \left| \frac{\partial \rho}{\partial \boldsymbol{z}}(\boldsymbol{z}) \right| \approx \int_0^L \nabla \cdot F(\boldsymbol{z}(s))\, ds$$

where equality would be achieved if ρ solves the dynamics exactly. But this divergence can be computed as follows:

$$\begin{aligned} \nabla \cdot F(\boldsymbol{z}(s)) &= \nabla_{\boldsymbol{x}} \cdot (\boldsymbol{u}) + \nabla_{\boldsymbol{u}} \cdot \left(-\frac{I - \boldsymbol{u}\boldsymbol{u}^T}{d-1} \nabla \mathcal{L}(\boldsymbol{x})\right) \\ &= 0 - \frac{1}{d-1} \nabla_{\boldsymbol{u}} \cdot \left((I - \boldsymbol{u}\boldsymbol{u}^T) \nabla \mathcal{L}(\boldsymbol{x})\right) \\ &= -\frac{1}{d-1} \boldsymbol{u}^T \nabla \mathcal{L}(\boldsymbol{x}). \end{aligned}$$

$W(\boldsymbol{z}', \boldsymbol{z})$ from the acceptance probability $\min(1, e^{-W(\boldsymbol{z}', \boldsymbol{z})})$ is therefore:

$$W(\boldsymbol{z}', \boldsymbol{z}) = \mathcal{L}(\boldsymbol{x}') - \mathcal{L}(\boldsymbol{x}) + \int_0^L \frac{1}{d-1} \boldsymbol{u}(s)^T \nabla \mathcal{L}(\boldsymbol{x}(s))\, ds$$

this integral is zero for the canonical case as ρ is simpletic but accounts for volume compression or expansion in the microcanonical case. $\boldsymbol{u}(s)^T \nabla \mathcal{L}(\boldsymbol{x}(s))$ measures the alignment between the log-density and the momentum.

As ρ approximates these dynamics using discrete steps, we can calculate [3] $W(\rho(\boldsymbol{z}), \boldsymbol{z})$ as:
$W(\rho(\boldsymbol{z}), \boldsymbol{z}) = \sum_{k=1}^{n} [V(A_{\epsilon_k}(\boldsymbol{z}_{k-1})) - V(\boldsymbol{z}_{k-1}) + K(B_{\epsilon_k} \circ A_{\epsilon_k}(\boldsymbol{z}_{k-1})) - K(A_{\epsilon_k}(\boldsymbol{z}_{k-1}))]$.

The potential energy change is the same as for canonical HMC: $V(A_\epsilon(\boldsymbol{z})) - V(\boldsymbol{z}) = -\log \frac{p(\boldsymbol{x}')}{p(\boldsymbol{x})}$. For canonical HMC with step size ϵ, the kinetic energy change is: $K(B_\epsilon(\boldsymbol{z})) - K(\boldsymbol{z}) = \frac{1}{2}\|\boldsymbol{u}'\|^2 - \frac{1}{2}\|\boldsymbol{u}\|^2$. For MAMS: $K(B_\epsilon(\boldsymbol{z})) - K(\boldsymbol{z}) = (d-1)\log(\cosh\delta + \boldsymbol{e}\cdot\boldsymbol{u}\sinh\delta)$ where $\boldsymbol{e} = -\nabla\mathcal{L}(\boldsymbol{x})/\|\nabla\mathcal{L}(\boldsymbol{x})\|$ and $\delta = \epsilon\|\nabla\mathcal{L}(\boldsymbol{x})\|/(d-1)$. Which was found by solving a nonlinear first order differential equation [2, 3, 12]. So, for both HMC and MAMS, the Metropolis-Hastings acceptance probability can be computed by tracking the total energy change W along the discretized trajectory. The crucial difference is that MAMS requires the volume correction term from the divergence of the velocity update, while HMC using canonical dynamics does not. The derivation is mathematically sound and resolves the limitation that prior microcanonical samplers (MCHMC, MCLMC) were asymptotically biased.

4 Behaviour of microcanonical dynamics

A key property of microcanonical dynamics emphasized by RCS is stability to large gradients. To understand this property, consider the behavior of particles under both systems of differential equations along a trajectory without momentum resampling.

The canonical dynamics $\dot{\boldsymbol{x}} = \boldsymbol{u}, \dot{\boldsymbol{u}} = -\nabla\mathcal{L}(\boldsymbol{x})$ leave momentum magnitude $\|\boldsymbol{u}\|$ unconstrained and varying according to the gradient. When $\|\nabla\mathcal{L}(\boldsymbol{x})\|$ is large, the update $\boldsymbol{u}_{k+1} = \boldsymbol{u}_k - \epsilon\nabla\mathcal{L}(\boldsymbol{x}_k)$ can produce $\|\boldsymbol{u}_{k+1}\| \gg \|\boldsymbol{u}_k\|$, leading to very large position steps $\boldsymbol{x}_{k+2} = \boldsymbol{x}_{k+1} + \epsilon\boldsymbol{u}_{k+1}$. This potentially causes rejection of the proposed state or numerical instability.

The microcanonical dynamics $\dot{\boldsymbol{x}} = \boldsymbol{u}, \dot{\boldsymbol{u}} = -\frac{I - \boldsymbol{u}\boldsymbol{u}^T}{d-1}\nabla\mathcal{L}(\boldsymbol{x})$ enforce the constraint $\|\boldsymbol{u}\| = 1$, implying $\|\dot{\boldsymbol{x}}\| = \|\boldsymbol{u}\| = 1$ always. The projection operator ensures $\dot{\boldsymbol{u}} \perp \boldsymbol{u}$, so momentum changes alter the direction, but not magnitude of $\boldsymbol{u}$. Even when $\|\nabla\mathcal{L}(\boldsymbol{x})\|$ is large, the speed remains constant. Large gradients cannot cause unbounded momentum growth.

The momentum restriction has implications for the characteristics of the particle's phase-space exploration.

For canonical dynamics, energy conservation in the Hamiltonian $H(\boldsymbol{x}, \boldsymbol{u}) = V(\boldsymbol{x}) + K(\boldsymbol{u})$ creates an inverse relationship between kinetic and potential energy. In high-probability regions, $p(\boldsymbol{x})$ is high and thus $\mathcal{L}(\boldsymbol{x}) = -\log p(\boldsymbol{x})$ is low. Potential energy is low and so kinetic energy must be correspondingly high. Particles move rapidly through high-density regions. Conversely, in low-probability regions with high potential energy, kinetic energy is low and particles may struggle to traverse these areas.

For microcanonical dynamics, the momentum magnitude is independent of the density at any point, ensuring a consistent speed of exploration through all regions. In steep regions, the particle's direction changes sharply due to the projection operator but speed remains constant. The Metropolis step can reject proposals to lower-density regions, preventing excessive exploration there. A potential issue arises when a ridge divides a flat region between two modes. Upon encountering such a ridge, the particle's momentum rotates but may fail to traverse the ridge, possibly causing the particle to fail to explore the full space.

5 Limitations

MAMS represents a technically correct algorithmic development that extends MCHMC and MCLMC with a Metropolis-Hastings correction, eliminating asymptotic bias through derivation of the acceptance probability for non-volume-preserving microcanonical dynamics. However, substantial theoretical and empirical gaps remain.

MAMS lacks convergence rate bounds for classes of target densities. For instance, canonical HMC has mixing time bounds for strongly log-concave targets [17] but no analogous results exist for MAMS. Without such an analysis, it is unclear when MAMS is guaranteed to converge faster than canonical methods and how performance scales with the dimension of the problem.

The paper provides no characterization of when the unit-norm constraint helps versus hinders exploration, nor diagnostic tools for MAMS-specific failure modes such as when volume corrections dominate acceptance decisions. It remains unclear for which target geometries favour microcanonical dynamics. In addition, MAMS exceeds standard HMC leapfrog implementation complexity as it requires computing the volume change.

In the case $d = 1$, the constraint that $\|\boldsymbol{u}\| = 1$ reduces to $\boldsymbol{u} \in \{-1, 1\}$. The projection operator $\frac{I-\boldsymbol{u}\boldsymbol{u}^T}{d-1}$ becomes undefined due to division by $d-1 = 0$. Regardless, in one dimension $I - \boldsymbol{u}\boldsymbol{u}^T = 0$, so the momentum never changes. Ver Steeg and Galstyan [12] noted this but dismissed it as the issue does not occur for higher dimensions.

The term 'Hamiltonian Monte Carlo' refers to methods based on Hamiltonian mechanics with volume-preserving dynamics [1, 4]. MAMS does not employ true Hamiltonian dynamics as noted by RCS [3]. RCS also note that the 'kinetic energy' for microcanonical dynamics is not in the standard sense. Drawing parallels to Hamiltonian dynamics and kinetic energy can confuse readers and practitioners.

The manuscript suffers from several presentation issues that detract from its technical contributions. Notation is inconsistent, with variables appearing in both script and non-script forms. The reference list contains duplicate entries with varying capitalization. More concerning is the self-citation. The manuscript exists as two arXiv versions (v1 from March 2025, v2 from May 2025, both arXiv:2503.01707), yet version 2 cites version 1.

6 Conclusion

The Metropolis-adjusted microcanonical sampler makes a technical contribution by deriving the acceptance probability for non-volume-preserving microcanonical dynamics, correcting the asymptotic bias present in MCHMC and MCLMC. The derivation is mathematically correct and demonstrates that Metropolis correction can be applied to dynamics whose integrators do not preserve phase-space volume. The unit-norm momentum constraint provides stability to large gradients, a property that may be of benefit in some cases.

A theoretical analysis of when and why microcanonical dynamics outperform canonical dynamics would be beneficial for this new methodology to gain trust and support. Additionally, presentation issues and misleading nomenclature reduce the work's accessibility.

7 Project: Comparison of MAMS Hyperparameter Tuning Schemes

7.1 Introduction

The Metropolis-adjusted microcanonical sampler (MAMS) algorithm requires careful tuning of two key hyperparameters: the trajectory length L, and the step size ϵ. While it is also possible to tune a preconditioner matrix, we simplify our analysis by setting it equal to the identity matrix. We focus on the MCHMC variant of MAMS, which samples a new unit-norm momentum vector between trajectories, but does not partially refresh the momentum within the trajectory, as would be done in the MCLMC variant.

The approach to tuning MAMS proposed by RCS [3] is a sequential optimization that tunes hyperparameters one at a time across three distinct phases, each of length $0.1N$ samples where N is the desired chain length. However, this sequential approach has a fundamental limitation: it cannot capture interactions between L and ϵ as each parameter is optimized while holding the other fixed. When ϵ is optimized with L held constant, then L is optimized with ϵ fixed, the resulting configuration may represent only a local optima.

We hypothesize that jointly exploring the (L, ϵ) space using Bayesian optimization (BayesOpt) will yield hyperparameter pairs closer to the global optimum. Instead of fixing one parameter while tuning the other, BayesOpt proposes candidate pairs from the hyperparameter space, evaluates their performance, updates its acquisition function based on the result, and iterates. After a chosen number of iterations, the hyperparameters which maximized a carefully chosen objective function are selected and used in all further chains.

For every chain, sequential tuning requires and discards an additional 20% of the desired chain length (or 30% when also tuning the preconditioner matrix) for the chain's idependent hyperparameter adaptation. If there is variability between hyperparamaters found for each chain, this suggests that the RCS tuning method converged to local optima. In contrast, BayesOpt invests heavily in an initial search phase but then reuses the optimal (L^*, ϵ^*) pair for all subsequent chains, effectively amortizing the initial computational cost. While this method may be more computationally expensive for small sampling projects, for large projects, the investment to find a potentially superior configuration without discarding a tuning phase for each chain may be beneficial. BayesOpt produces a single set of hyperparameters for all future chains, whereas auto-tuning produces different hyperparamater for each chain. BayesOpt explores the (L, ϵ) space jointly, capturing potential interactions that sequential tuning cannot detect. Our hypothesis is that joint optimization of (L, ϵ) through BayesOpt should discover better configurations with more consistent behaviour across chains.

We evaluate both methods on three target densities: the banana distribution, a correlated Gaussian, and an equal mixture of two Gaussians. The correlated Gaussian and bimodal Gaussian are specifically constructed to have equal marginal variance, so diagonal preconditioning ought to be ineffective. We implement MAMS using the BlackJAX [18] library in JAX [19]. While the implementation of MAMS appears correct, it was necessary to modify the adaptation function to enable fixing L during the dual averaging adaptation of ϵ, as this functionality was not accessible by default. The adaptation function had to be duplicated and modified to allow the sequential tuning protocol described by RCS.

The code to reproduce the experiments is available on GitHub:
https://github.com/1saacRankin/QP2-microcanonical-HMC.

7.2 Sequential Auto-Tuning Method

The sequential tuning approach, as described by RCS in their manuscript [3], adapts hyperparameters across three stages. The algorithm is initialized with $L = \sqrt{d}$ where d is the dimension of the target density, and sets the inverse mass matrix to the identity matrix $\boldsymbol{M}^{-1} = \boldsymbol{I}$. In Stage 1, which runs for $0.10N$ steps, dual averaging tunes ϵ to target a 90% acceptance rate while holding L fixed. Stage 2, also running for $0.10N$ steps, estimates a diagonal preconditioning matrix (though we skipped this stage in our experiments). Stage 3 refines L based on autocorrelation time estimates while holding ϵ fixed, again for $0.10N$ steps.

After these tuning phases, the algorithm runs a chain of length N using the tuned (L, ϵ) pair. A critical

feature of this approach is that each chain runs this tuning procedure independently, producing its own hyperparameters. This means that chains may converge to different (L, ϵ) pairs, which could indicate either a broad range of hyperparamters are effective, or that there is instability in the tuning procedure.

Note: as the chain automatically produces hyperparameters for itself, from the word automatically and the Greek prefix for self, we shall refer to this tuning method as 'Auto-tuning'.

7.3 Bayesian Optimization Method

Our BayesOpt approach jointly optimizes (L, ϵ) by modeling sampler performance as a function of both hyperparameters simultaneously. We use a Gaussian process (GP) with a radial basis function (RBF) kernel as our surrogate model, and use the upper confidence bound (UCB) acquisition function to balance exploration and exploitation. The search proceeds by iteratively proposing hyperparameter pairs, evaluating their performance across multiple chains, and updating the GP model based on the observed results.

Algorithm 1 Bayesian Optimization for MAMS

Input: Target density, iterations $T = 20$, chains per evaluation $C = 5$, chain length $N_{\text{tune}} = 1000$
Initialize search space: $L \in [0.5, 40.0]$, $\epsilon \in [0.01, 4.0]$
Initialize Gaussian process with RBF kernel
for $t = 1$ to T **do**
  Select (L_t, ϵ_t) by maximizing $\text{UCB}(L, \epsilon) = \mu(L, \epsilon) + 2 \cdot \sigma(L, \epsilon)$
  Run C independent chains of length N_{tune} with hyperparameters (L_t, ϵ_t)
  Compute minimum ESS across dimensions for each chain, then average over the C chains
  Compute maximum $\hat{R}$ across dimensions for each chain, then average over the C chains
  Compute mean acceptance rate α for each chain, then average over the C chains
  Evaluate objective: $f_t(L_t, \epsilon_t) = \text{ESS}(L, \epsilon) - P_{\hat{R}}(L_t, \epsilon_t) - P_\alpha(L_t, \epsilon_t)$
  Update GP with observation (L_t, ϵ_t, f_t)
end for
Return: Hyperparameters $(L^*, \epsilon^*) = (L_t, \epsilon_t)$ where $t = \arg\max_t f_t$

The objective function aims to maximize effective sample size (ESS) while heavily penalizing configurations that lead to non-convergent chains or artificially inflated ESS due to extremely low acceptance rates:

$$f(L, \epsilon) = \text{ESS}(L, \epsilon) - P_{\hat{R}}(L, \epsilon) - P_\alpha(L, \epsilon).$$

The convergence penalty $P_{\hat{R}}$ ensures that we only trust ESS estimates from chains that have actually converged according to $\hat{R}$ [20]:

$$P_{\hat{R}}(L, \epsilon) = C^2 \cdot N_{\text{tune}} \cdot \max(\hat{R}(L, \epsilon) - 1.0, 0).$$

The acceptance penalty P_α prevents high ESS values that can arise when acceptance rates are extremely low. When the sampler rarely moves, the samples can have low autocorrelation, and so high ESS. To avoid this illusion of good performance, we linearly decrease the acceptance penalty as acceptance rates increase from 0 to 0.25:

$$P_\alpha(L, \epsilon) = \begin{cases} C^2 \cdot N_{\text{tune}} \cdot (0.25 - \alpha(L, \epsilon)) & \text{if } \alpha < 0.25 \\ 0 & \text{otherwise} \end{cases}$$

After 20 iterations, each using 5 chains of length 1000, that is, 100000 samples, we select the hyperparameters that achieved the highest objective value. These optimal hyperparameters (L^*, ϵ^*) are then used for all subsequent validation chains.

7.4 Validation Protocol

We validate both methods using 10 independent chains of length 5000, each initialized from a different random location. For BayesOpt, all chains use the single hyperparameter pair (L^*, ϵ^*) identified during the tuning phase. For sequential auto-tuning, each chain independently discovers its own pair (L_i^*, ϵ_i^*) for $i = 1, 2, \ldots, 10$, following the adaptation protocol described before.

We compare the methods across several key metrics. The minimum effective sample size across all dimensions measures sampling efficiency [21]. The maximum $\hat{R}$ across dimensions assesses convergence, with values near 1.0 indicating more likely convergence. We also track the mean acceptance rate and wall-clock time per chain. For the 10 validation chains, we present $\hat{R}$, and the mean and standard deviation of ESS, acceptance rate, and time per chain. For Auto-tuned chains, we additionally examine the variability in the L and ϵ values across chains to assess the consistency of the tuning procedure.

7.5 Bimodal Isotropic Gaussian

The bimodal isotropic Gaussian target consists of two separated modes with equal weight and identity covariance matrices. The density is defined as an equal mixture:

$$p(\boldsymbol{x}) = \frac{1}{2}\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_1, \boldsymbol{I}) + \frac{1}{2}\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_2, \boldsymbol{I})$$

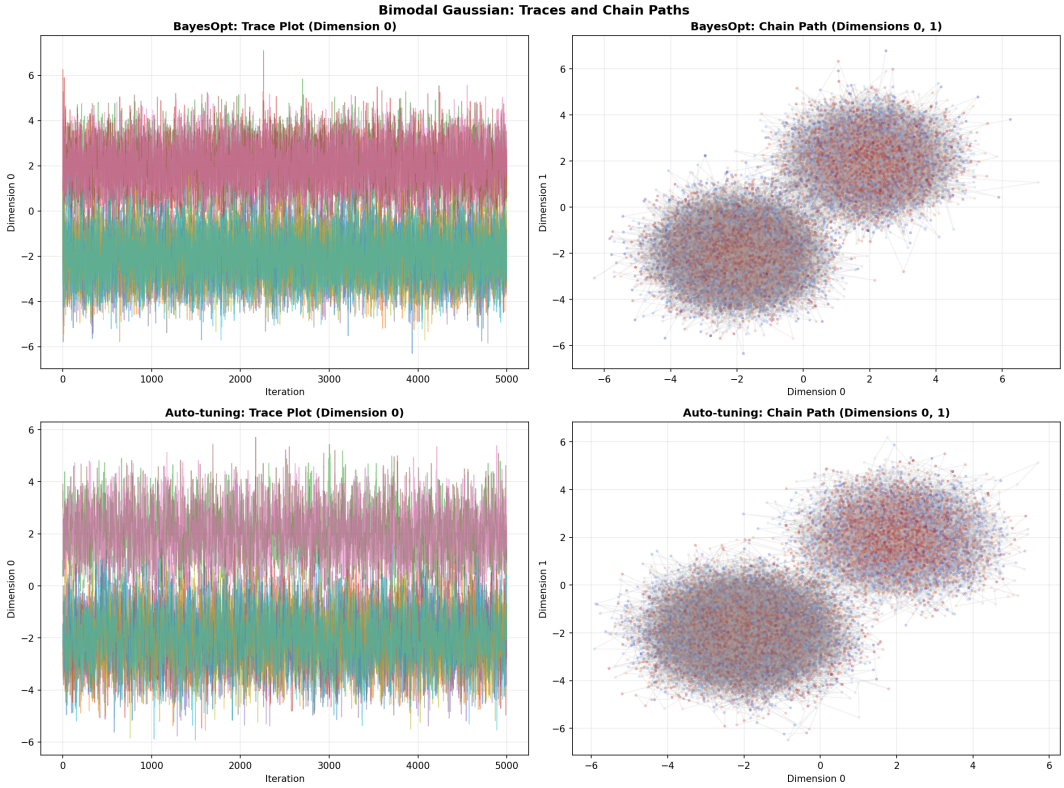
where $\boldsymbol{\mu}_1 = [-2, -2, \ldots, -2]^T$ and $\boldsymbol{\mu}_2 = [2, 2, \ldots, 2]^T$ in $d = 10$ dimensions. The log density is proportional to:

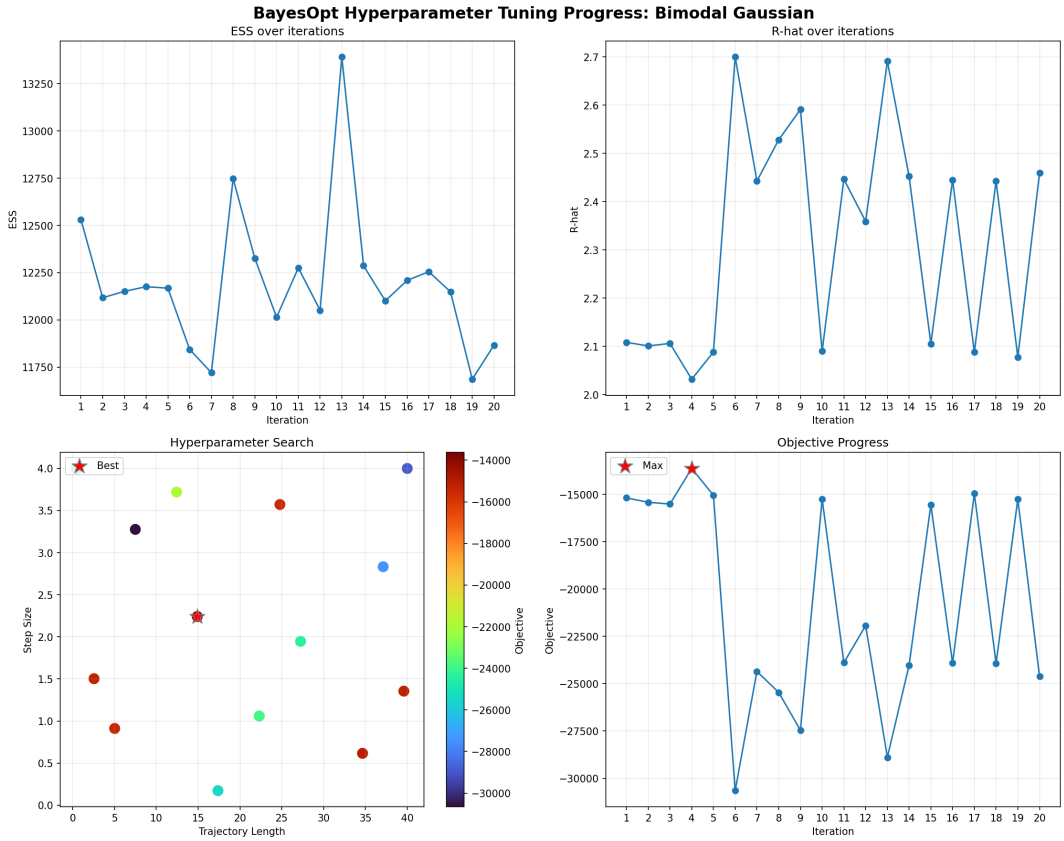
$$\log p(\boldsymbol{x}) \propto -\frac{1}{2}\|\boldsymbol{x} - \boldsymbol{\mu}_1\|^2 - \frac{1}{2}\|\boldsymbol{x} - \boldsymbol{\mu}_2\|^2$$

This target tests the sampler's ability to switch between modes and properly explore both regions of high probability. All dimensions have symmetric structure and are independent within each mode, so diagonal preconditioning offers no advantage over an identity matrix. The challenge of this target density is achieving adequate mixing between the two modes.

Table 1: Performance comparison on Bimodal Gaussian ($d = 10$)

Method	$\hat{R}$	ESS	Acceptance Rate	Time/chain (s)	L	ϵ
BayesOpt	2.300	61023 ± 778	0.964 ± 0.001	0.742 ± 0.167	14.901	2.242
Auto-tuning	2.189	61091 ± 1129	0.988 ± 0.003	2.908 ± 0.693	1.310 ± 0.119	1.191 ± 0.108


Bimodal Gaussian: Traces and Chain Paths
BayesOpt: Trace Plot (Dimension 0)
BayesOpt: Chain Path (Dimensions 0, 1)
Auto-tuning: Trace Plot (Dimension 0)
Auto-tuning: Chain Path (Dimensions 0, 1)
Dimension 0
Dimension 1
Iteration


BayesOpt Hyperparameter Tuning Progress: Bimodal Gaussian
ESS over iterations
R-hat over iterations
Hyperparameter Search
Objective Progress
Best
Max
ESS
R-hat
Step Size
Trajectory Length
Objective
Iteration

7.6 Correlated Gaussian

The correlated Gaussian target is a centered at the origin with a specific correlation structure:

$$p(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x}|\mathbf{0}, \boldsymbol{\Sigma})$$

where the covariance matrix has constant correlation $\rho = 0.8$:

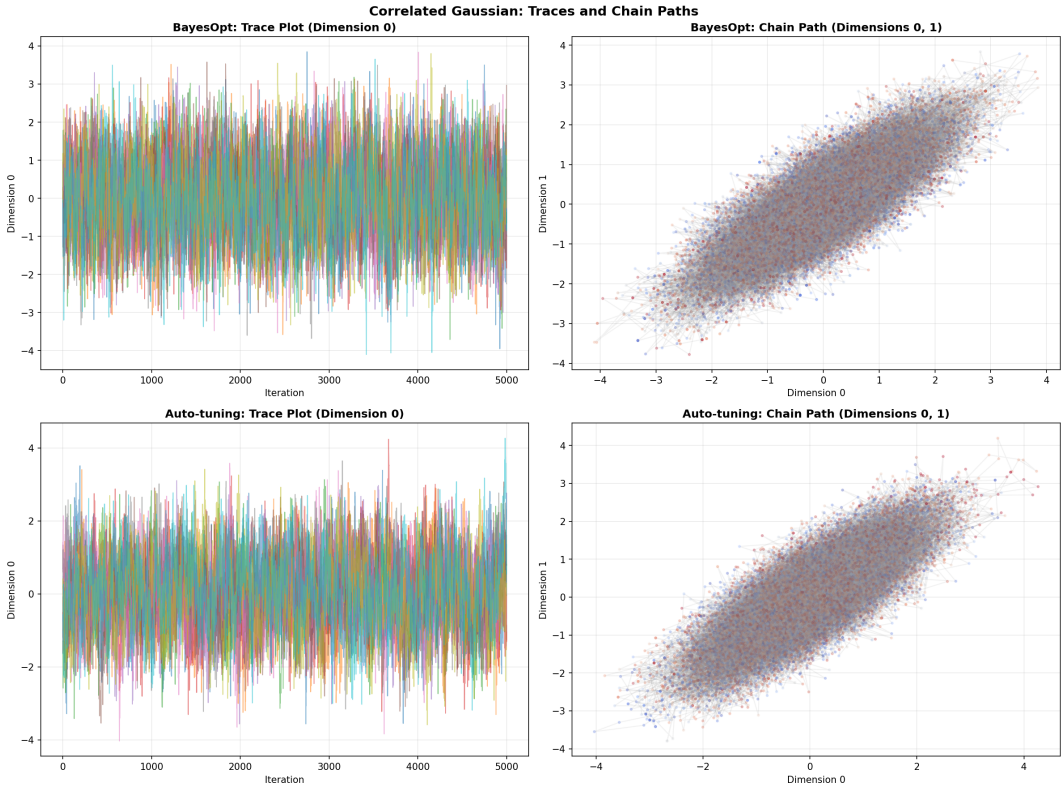
$$\boldsymbol{\Sigma} = (1-\rho)\boldsymbol{I} + \rho\mathbf{1}\mathbf{1}^T$$

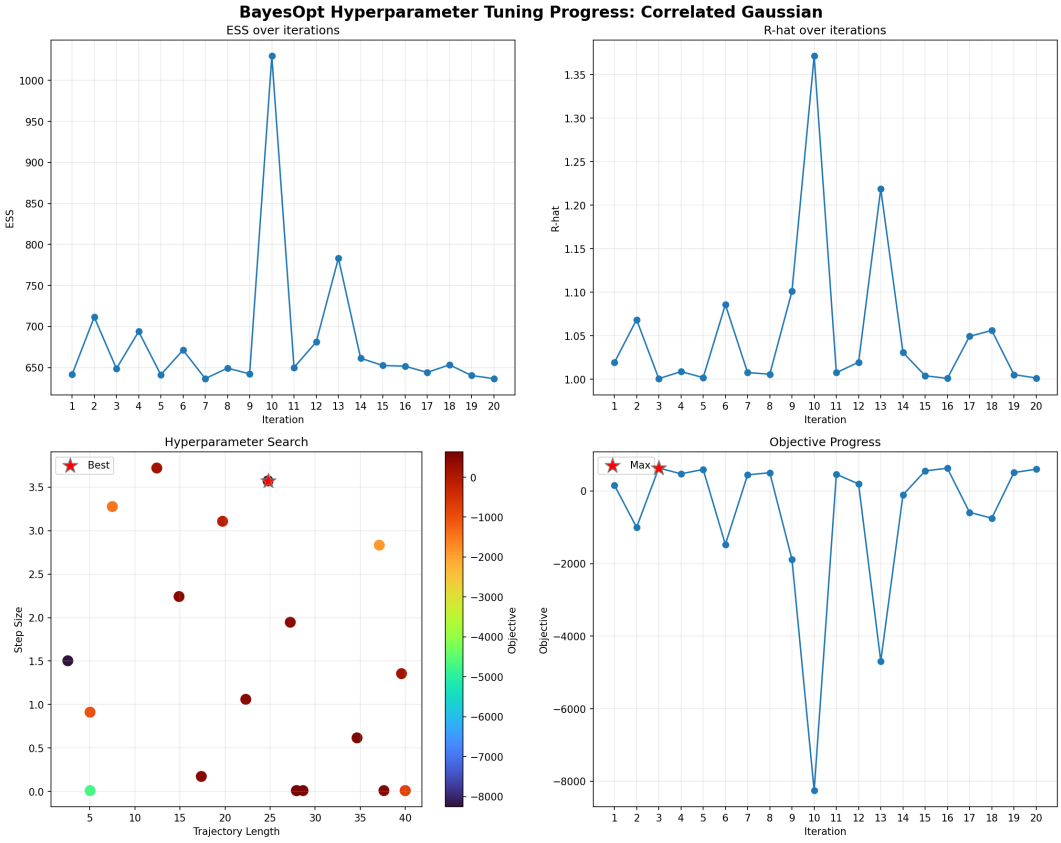
This covariance matrix has ones on the diagonal and $\rho = 0.8$ in all off-diagonal entries. Notice that a diagonal preconditioning cannot capture the off-diagonal correlation structure. The log density is:

$$\log p(\boldsymbol{x}) \propto -\frac{1}{2}\boldsymbol{x}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{x}$$

Table 2: Performance comparison on Correlated Gaussian ($d = 50$)

Method	$\hat{R}$	ESS	Acceptance Rate	Time/chain (s)	L	ϵ
BayesOpt	1.003	3186 ± 31	0.756 ± 0.003	0.763 ± 0.155	24.776	3.572
Auto-tuning	1.003	3223 ± 34	0.891 ± 0.031	3.590 ± 0.781	14.142 ± 0.000	2.650 ± 0.085





7.7 Banana Distribution

The banana distribution, also known as the Rosenbrock or crescent distribution, has a curved shape that can be a significant challenge for samplers. In d dimensions, the density is:

$$p(\boldsymbol{x}) \propto \exp\left(-\sum_{i=1}^{d-1}\left[(a-x_i)^2 + b(x_{i+1}-x_i^2)^2\right]\right).$$

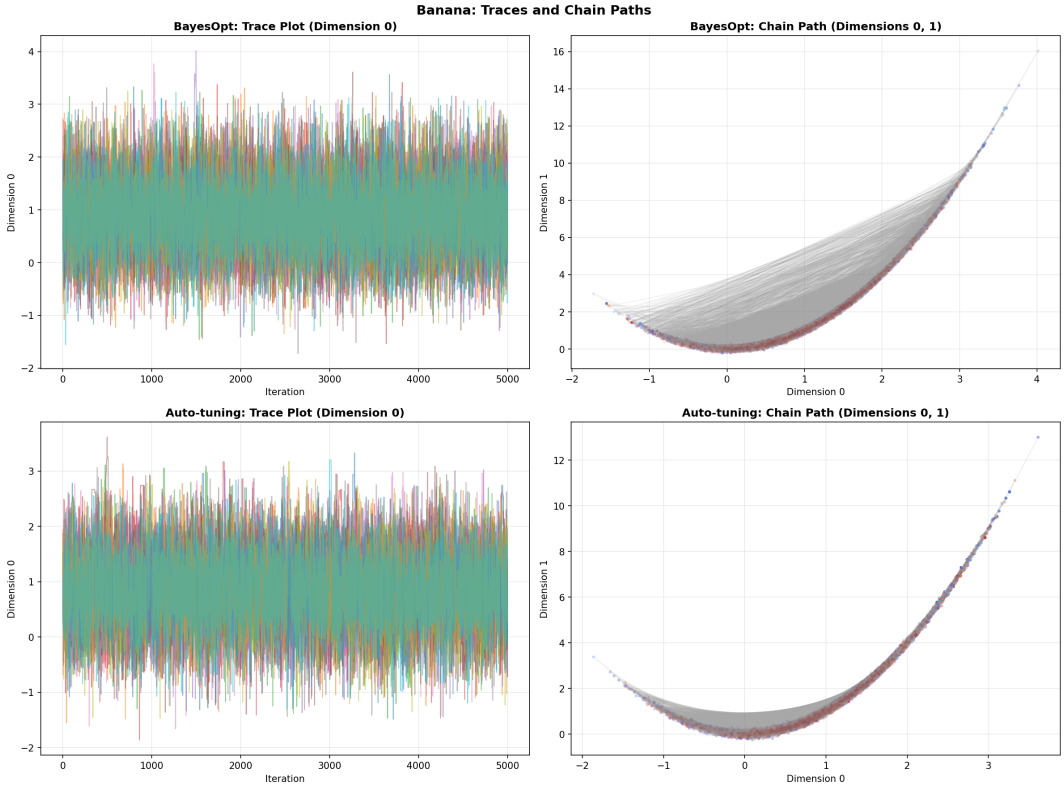
When $d = 2$ with parameters $a = 1$ and $b = 100$, the log density becomes:

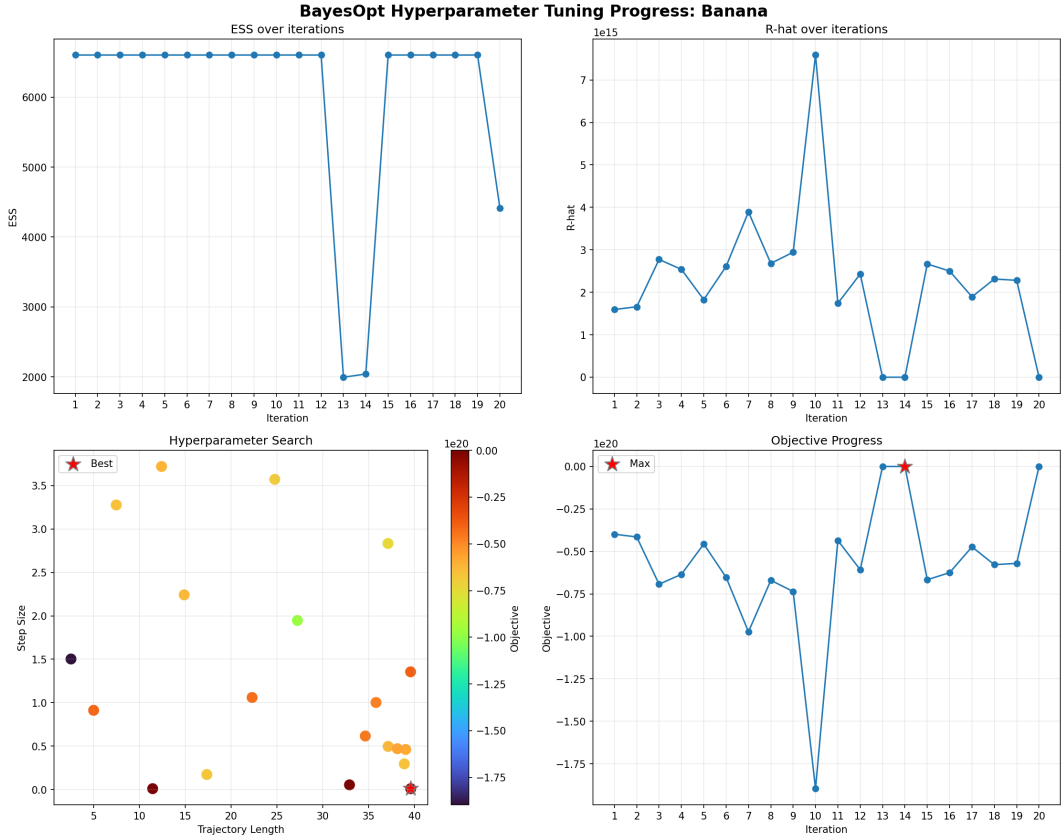
$$\log p(\boldsymbol{x}) \propto -(1-x_1)^2 - 100(x_2 - x_1^2)^2$$

This distribution creates strong nonlinear dependencies between consecutive dimensions. The high curvature results in a narrow curved ridge with the probability mass. This is a difficult target density for samplers, and its geometry is a different challenge than either the multimodal or correlated Gaussian targets.

Table 3: Performance comparison on Banana Distribution ($d = 2$)

Method	$\hat{R}$	ESS	Acceptance Rate	Time/chain (s)	L	ϵ
BayesOpt	1.000	10132 ± 243	0.946 ± 0.003	37.839 ± 0.275	39.602	0.010
Auto-tuning	1.001	10182 ± 460	0.885 ± 0.027	9.114 ± 1.562	2.828 ± 0.000	0.020 ± 0.002


Banana: Traces and Chain Paths
BayesOpt: Trace Plot (Dimension 0)
BayesOpt: Chain Path (Dimensions 0, 1)
Auto-tuning: Trace Plot (Dimension 0)
Auto-tuning: Chain Path (Dimensions 0, 1)
Dimension 0
Dimension 1
Iteration


BayesOpt Hyperparameter Tuning Progress: Banana
ESS over iterations
R-hat over iterations
Hyperparameter Search
Objective Progress
ESS
R-hat
Step Size
Trajectory Length
Objective
Iteration
Best
Max

7.8 Discussion

For all three target distributions, both BayesOpt and Auto-tuning hyperparameter tuning methods produced similar sampling performances. This suggests that for these targets, the (L, ϵ) hyperparameter space contains regions where hyperparamater configurations perform similarly, and so these targets do not require precise tuning.

We can examine the EEE, $\hat{R}$, and objective as BayesOpt iteratively explores the hyperparameter space. For the bimodal Gaussian, as the chains never mixed well, the best hyperparamaters appear to have minimized $\hat{R}$. For none of the three targets do we see the hyperparameters which maximize the objective function occuring near the end of the chain. Perhaps if BayesOpt was granted more iterations we would see more optimization rather than exploration.

For each target and each tuning method, we plot the trace of the chain for its first dimension to assess convergence. The bimodal target has clear poor mixing. We also show a scatterplot of the first two dimensions. For the banana, we can see that the BayesOpt tuning allows for much greater distances between consecutive samples.

For the bimodal Gaussian, both methods fail to achieve adequate convergence, with $\hat{R} > 2$. Neither method found hyperparameters capable of easily mode-switching in the validation chain length of 5000. The mean ESS values are nearly identical (61023 for BayesOpt versus 61091 for auto-tuning), which are much greater than the chain length of 5000, and both methods have high acceptance rates above 0.96. The notable difference lies in the discovered hyperparameters: BayesOpt selected a much longer trajectory length of 14.901 paired with a larger step size of 2.242 compared to those of Auto-tuning with $L = 1.310 \pm 0.119$ and $\epsilon = 1.191 \pm 0.108$. Notice that $\lceil 1.310/1.191 \rceil = 2$ and so the trajectory simulated is very short. Here, the standard deviations of L and ϵ between chains is not negligible, though it is possible that $\lceil L\epsilon \rceil = n = 2$ holds for all 10 chains. BayesOpt achieves substantially faster wall-clock time (0.742 seconds versus 2.908 seconds), likely due to having pre-optimized hyperparameters.

Both methods succesfuly converge for the correlated Gaussian, both with $\hat{R} = 1.003$. Here again, ESS values are nearly identical (3186 for BayesOpt versus 3223 for auto-tuning). Again, BayesOpt chooses a longer trajectory and larger step size, 24.776 versus 14.142, and 3.572 versus 2.650 respectively. The auto-tuning method achieves higher acceptance rates (0.891 versus 0.756). BayesOpt performed better in terms of wall-clock time, completing chains in 0.763 seconds compared to 3.590 seconds for auto-tuning.

For the banana distribution, both methods achieve excellent convergence with $\hat{R} \approx 1.000$. Both methods have nearly identical ESS values (10132 for BayesOpt versus 10182 for auto-tuning). The hyperparameter differences are extreme. BayesOpt selected an extremely long trajectory $L = 39.602$ with a very small step size $\epsilon = 0.010$. These are nearly at the bounds of the hyperparameter search space with $L \in [0.5, 40.0]$ and $\epsilon \in [0.01, 4.0]$. Auto-tuning chose much shorter trajectories $L = 2.828$ with slightly larger step sizes $\epsilon = 0.020$, both with small variability. These represent fundamentally different strategies with BayesOpt choosing many small steps along a long trajectory whereas Auto-tuning chose fewer larger steps along a short trajectory. And interestingly, BayesOpt with the lower trajectory had higher acceptance rates. The wall-clock time relationship seen for the bimodal Gaussian and correlated Gaussian reverses for this target, with auto-tuning completing chains in 9.114 ± 1.562 seconds compared to BayesOpt's 37.839 ± 0.275 seconds. This is not surprising as the BayesOpt trajectory requires many more computations than that of auto-tuning.

Interestingly, auto-tuning discovers hyperparameters with low variability across chains, suggesting this hyperparameter tuning algorithm converges to a local optima.

In our experiments, auto-tuning was slower for the bimodal and correlated Gaussian targets. For the banana distribution, BayesOpt's extremely long trajectories made it slower.

For computational cost, for each target here, BayesOpt spends 100000 samples upfront but reuses the same hyperparameters for all chains. Auto-tuning discarded 1000 samples adapting each of the 10 chains of length 5000, discarding 10000 samples total for the validation portion of the analysis. For large-scale applications with many chains, BayesOpt's amortized cost may be lower. That said, you could also share hyperparameters

found through auto-tuning across chains.

Our approach has several limitations. We only tune trajectory length and step size, leaving the inverse mass matrix fixed. We do not consider the Langevin-like version of MAMS, which has an additional partial momentum resampling hyperparameter. The BayesOpt search uses 20 iterations, which may be insufficient for some targets and hyperparameter search space. The hyperparamater search space used here allows for the step size to be greater than the trajectory length, though this did not become a problem, it should be fixed in the future. Our objective function, lacks rigorous theoretical justification. It was constructed to that way to avoid high ESS from poor mixing and many rejections. Here, we have not compared either tuning method to a canonical HMC algorithm, and so it is possible that the sampler can be outperformed regardless of the tuning method. Finally, our experiments are limited to three relatively low-dimensional targets with $d \leq 50$. Performance in higher dimensions remains untested.

7.9 Conclusion

We compared the sequential tuning algorithm of RCS to joint optimization via Bayesian optimization. Our results indicate that the choice of tuning method has minimal impact on sampling performance. The two methods choose drastically different trajectory lengths and step sizes. Despite discovering substantially different hyperparameter configurations, both methods achieve similar effective sample sizes and $\hat{R}$ values across all three targets tested. This means one can choose a tuning method based on convenience and computational budget. BayesOpt requires a one-time upfront optimization, auto-tuning requires adaptation every per chain. Which is preferable depends on which discards fewer samples.

References

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

[2] Jakob Robnik, G Bruno De Luca, Eva Silverstein, and Uroš Seljak. Microcanonical Hamiltonian Monte Carlo. *Journal of Machine Learning Research*, 24(311):1–34, 2023.

[3] Jakob Robnik, Reuben Cohn-Gordon, and Uroš Seljak. Metropolis Adjusted Microcanonical Hamiltonian Monte Carlo. *arXiv preprint arXiv:2503.01707*, 2025.

[4] Michael Betancourt. A conceptual introduction to Hamiltonian Monte Carlo. *arXiv preprint arXiv:1701.02434*, 2017.

[5] Jakob Robnik and Uroš Seljak. Fluctuation without dissipation: Microcanonical Langevin Monte Carlo. *arXiv preprint arXiv:2303.18221*, 2023.

[6] Jakob Robnik, Reuben Cohn-Gordon, and Uroš Seljak. Black-box unadjusted Hamiltonian Monte Carlo. *arXiv preprint arXiv:2412.08876*, 2024.

[7] David Tong. Statistical Physics University of Cambridge Part II Mathematical Tripos. *arXiv preprint arXiv:0908.0333*, page 52, 2009.

[8] 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.

[9] Xiaoyu Lu, Valerio Perrone, Leonard Hasenclever, Yee Whye Teh, and Sebastian Vollmer. Relativistic Monte Carlo. In *Artificial Intelligence and Statistics*, pages 1236–1245. PMLR, 2017.

[10] Nilesh Tripuraneni, Mark Rowland, Zoubin Ghahramani, and Richard Turner. Magnetic Hamiltonian Monte Carlo. In *International conference on machine learning*, pages 3453–3461. PMLR, 2017.

[11] Ari Pakman. Super-efficient exact Hamiltonian Monte Carlo for the von Mises distribution. *Applied Mathematics Letters*, 159:109284, 2025.

[12] Greg Ver Steeg and Aram Galstyan. Hamiltonian Dynamics with Non-Newtonian Momentum for Rapid Sampling. *Advances in Neural Information Processing Systems*, 34:11012–11025, 2021.

[13] Max Hird and Samuel Livingstone. Quantifying the effectiveness of linear preconditioning in Markov chain Monte Carlo. *Journal of Machine Learning Research*, 26(119):1–51, 2025.

[14] Andrew Campbell, Wenlong Chen, Vincent Stimper, Jose Miguel Hernandez-Lobato, and Yichuan Zhang. A gradient based strategy for Hamiltonian Monte Carlo hyperparameter optimization. In *International Conference on Machine Learning*, pages 1238–1248. PMLR, 2021.

[15] Matthew D Hoffman, Andrew Gelman, et al. The No-U-Turn sampler: adaptively setting path lengths in Hamiltonian Monte Carlo. *J. Mach. Learn. Res.*, 15(1):1593–1623, 2014.

[16] Yurii Nesterov. Primal-dual subgradient methods for convex problems. *Mathematical programming*, 120(1):221–259, 2009.

[17] Zongchen Chen and Santosh S Vempala. Optimal convergence rate of Hamiltonian Monte Carlo for strongly logconcave distributions. *arXiv preprint arXiv:1905.02313*, 2019.

[18] Alberto Cabezas, Adrien Corenflos, Junpeng Lao, Rémi Louf, Antoine Carnec, Kaustubh Chaudhari, Reuben Cohn-Gordon, Jeremie Coullon, Wei Deng, Sam Duffield, et al. BlackJAX: composable Bayesian inference in JAX. *arXiv preprint arXiv:2402.10797*, 2024.

[19] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, et al. Jax: Autograd and xla. *Astrophysics Source Code Library*, pages ascl–2111, 2021.

[20] Andrew Gelman and Donald B Rubin. Inference from iterative simulation using multiple sequences. *Statistical science*, 7(4):457–472, 1992.

[21] Aki Vehtari, Andrew Gelman, Daniel Simpson, Bob Carpenter, and Paul-Christian Bürkner. Rank-normalization, folding, and localization: An improved $\hat{R}$ for assessing convergence of mcmc (with discussion). *Bayesian Analysis*, 16(2):667–718, 2021.