Journal of Machine Learning Research 1 (2016) 1-48 Submitted 4/00; Published 10/00

Manifold Langevin Monte Carlo with Partial Metric Updates

Theodore Papamarkou THEODORE.PAPAMARKOU@GLASGOW.AC.UK
School of Mathematics and Statistics
University of Glasgow, UK

Eric Ford EBF11@PSU.EDU
Center for Astrostatistics
Pennsylvania State University, USA

Editor: Leslie Pack Kaelbling

Abstract

Manifold Markov chain Monte Carlo algorithms have been introduced to sample more effectively from challenging target densities exhibiting multiple modes or strong correlations. Such algorithms exploit the local geometry of the parameter space, thus simulating chains with high convergence rate. However, acquiring local geometric information increases computational complexity to the extent that sampling from high-dimensional targets becomes inefficient. This paper proposes a manifold Langevin Monte Carlo framework aimed at maintaining a feasible amount of geometric computation. The suggested strategy regulates the frequency of manifold-based updates via a schedule, hence it controls the trade-off between effective sample size and computational cost leading to improved computational efficiency. An exponentially decaying schedule is put forward that enables more frequent updates of geometric information in early transient phases of the chain, while saving computational time in late stationary phases. Alternatively, a modulo schedule is introduced to allow sparse yet recurring geometric updates throughout the sampling course, acting as a correction mechanism for adaptive proposals that learn the covariance structure of the parameter space. The average complexity can be manually set for either of these two schedules depending on the need for geometric exploitation posed by the underlying model.

Keywords: Bayesian inference, High-dimensional data analysis, Markov chain Monte Carlo, Langevin Monte Carlo

1. Introduction

The birth of geometric-based Markov chain Monte Carlo (MCMC) dates back to the landmark paper of Duane et al. (1987), which introduced Hamiltonian Monte Carlo (HMC) to unite MCMC with molecular dynamics. Duane et al. (1987) applied HMC in lattice field theory simulations of quantum chromodynamics. Statistical applications of HMC began with its use in neural network models by Neal (1996).

In the meanwhile, the Metropolis-adjusted Langevin algorithm (MALA) was proposed by Roberts and Rosenthal (1998) to harness Langevin dynamics for proposing new states in MCMC simulations. Both HMC and MALA evaluate the gradient of the target density, so they utilize local geometric flow.

The seminal work of Girolami and Calderhead (2011) raised awareness of the differential geometric foundations of MCMC. Given a vector of parameters $\boldsymbol{\theta} \in \mathbb{R}^{n_\theta}$, Girolami and

Calderhead (2011) defines a distance between two probability densities $p(\boldsymbol{\theta})$ and $p(\boldsymbol{\theta} + \delta\boldsymbol{\theta})$ as the quadratic form $\delta\boldsymbol{\theta}^T G(\boldsymbol{\theta})\delta\boldsymbol{\theta}$ for an arbitrary metric $G(\boldsymbol{\theta})$. Thus, the position-specific metric $G(\boldsymbol{\theta})$ induces a Riemann manifold in the space of parameterized probability density functions $\{p(\boldsymbol{\theta}) : \boldsymbol{\theta}\}$. In the context of MCMC, $G(\boldsymbol{\theta})$ depends on the current state $\boldsymbol{\theta}$ of the simulated Markov chain. Girolami and Calderhead (2011) uses $G(\boldsymbol{\theta})$ to define transition kernels that explore the parameter space $\{\boldsymbol{\theta} : \boldsymbol{\theta} \in \mathbb{R}^{n_\theta}\}$ effectively, thereby introducing Riemann manifold Langevin and Hamiltonian Monte Carlo methods.

The differential geometric basis of MCMC has been further researched since the appearance of Girolami and Calderhead (2011). For instance, Byrne and Girolami (2013) developed proposal mechanisms by following geodesic flows over the embedded manifolds of support for the target distribution. Lan et al. (2015) replaced the auxiliary momentum variable appearing in Riemann manifold Hamiltonian Monte Carlo (RMHMC) by velocity to develop a Markov chain Monte Carlo algorithm using Lagrangian dynamics. Lan et al. (2014) defined the so-called wormhole metric, for which distance between modes is shortened, in order to facilitate transitions between modes of multimodal target densities.

Computing the geometric entities of differential geometric MCMC methods creates a performance bottleneck that restricts the applicability of the involved methods. For example, manifold Monte Carlo methods require to calculate the metric tensor $G(\boldsymbol{\theta})$ of choice. Typically, $G(\boldsymbol{\theta})$ is set to be the observed Fisher information matrix, which equals the negative Hessian of the log-target density at a specific point $\boldsymbol{\theta}$. Consequently, the complexity of manifold MCMC algorithms is dominated by Hessian-related computations, including inversion of the Hessian, so it becomes at best of order $\mathcal{O}(n_\theta^3)$.

In spite of being of order $\mathcal{O}(n_\theta^3)$ too, the simplified manifold Metropolis-adjusted Langevin algorithm (SMMALA) introduced by Girolami and Calderhead (2011) is faster than MMALA and RMHMC in lower-dimensional parameter spaces. The relatively faster speed of SMMALA over MMALA and RMHMC is explained by lower order terms and constant factors appearing in big-oh notation, which are ordinarily omitted but affect runtime in the case of smaller n_θ. For this reason, the simplified manifold Metropolis adjusted Langevin algorithm (SMMALA) has been employed in conjunction with population MCMC for the Bayesian analysis of mechanistic models based on systems of non-linear differential equations, see Calderhead and Girolami (2011) and Schwentner et al. (2015). Despite the capacity of SMMALA to exploit local geometric information so as to cope with non-linear correlations and modest increase in the dimensionality of the parameter space, in other cases its computational complexity can render the performance inferior to other algorithms such as the Metropolis-adjusted Langevin algorithm (MALA) or adaptive MCMC, see Calderhead et al. (2013).

Various attempts have been made to ameliorate the computational implications of geometric Monte Carlo methods. Along these lines, Lan et al. (2016) used Gaussian processes to emulate the Hessian matrix and Christoffel symbols associated with the observed Fisher information $G(\boldsymbol{\theta})$. Şimşekli et al. (2016) developed a stochastic quasi-Newton Langevin MCMC algorithm which takes into account the local geometry, while approximating the inverse Hessian by using a limited history of samples and their gradients. Alternatively, Pereyra (2016) used convex analysis and proximal techniques instead of differential calculus in order to construct a Langevin Monte Carlo method for high-dimensional target distributions that are log-concave and possibly not continuously differentiable.

This paper develops a Monte Carlo method that capitalizes on the sampling effectiveness of SMMALA and at the same time breaks the computational barrier of $\mathcal{O}(n_\theta^3)$. The proposed method permits to adjust the schedule of metric updates. Moreover, it is simple to implement and it performs exact metric evaluations, avoiding emulation or other approximations of the Hessian of the log-target density.

2. Background

The role of this section is to provide a brief overview of Langevin Monte Carlo and to stage a unified notation, which will be used for introducing the method contributed by this paper.

From a statistical perspective, Langevin Monte Carlo is a special case of Metropolis-Hastings with a normal proposal distribution

$$q(\boldsymbol{\theta}^\star|\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta}^\star|\boldsymbol{\mu}(\boldsymbol{\theta}, G, \epsilon), \Sigma(G, \epsilon)), \tag{1}$$

where $\boldsymbol{\theta}^\star$ and $\boldsymbol{\theta}$ denote the respective proposed and current state, G is a positive definite matrix of size $n_\theta \times n_\theta$ and ϵ refers to a tuning parameter known as the integration stepsize. The location $\boldsymbol{\mu}(\boldsymbol{\theta}, G, \epsilon)$ is a function of $\boldsymbol{\theta}$, G and ϵ, whereas the covariance $\Sigma(G, \epsilon)$ of the proposal density depends on G and ϵ. Both $\boldsymbol{\mu}(\boldsymbol{\theta}, G, \epsilon)$ and $\Sigma(G, \epsilon)$ are defined so that the proposed states admit a Langevin diffusion approximated by a first-order Euler discretization. In other words, the Langevin diffusion has a stationary distribution $p(\boldsymbol{\theta})$, so a Markov chain simulated from the Metropolis-Hastings scheme outlined by proposal (1) will converge to the target $p(\boldsymbol{\theta})$. The Metropolis-Hastings acceptance probability is to its standard form

$$a(\boldsymbol{\theta}, \boldsymbol{\theta}^\star) = \min\left\{\frac{p(\boldsymbol{\theta}^\star)q(\boldsymbol{\theta}|\boldsymbol{\theta}^\star)}{p(\boldsymbol{\theta})q(\boldsymbol{\theta}^\star|\boldsymbol{\theta})}, 1\right\}. \tag{2}$$

The integration stepsize ϵ is associated with the first order Euler discretization and its value affects the state space exploration. If ϵ is selected to be relatively large, some of the proposed candidates will be far from the current state, and will therefore have a low probability of acceptance, so the generated chain will have low acceptance rate. Reducing ϵ will increase the acceptance rate, but the chain will take longer to traverse the state space.

In a Bayesian setting, the target is a possibly unnormalized posterior distribution $p(\boldsymbol{\theta}|\mathbf{y})$, where $\mathbf{y}$ denotes the available data. Replacing $p(\boldsymbol{\theta})$ by $p(\boldsymbol{\theta}|\mathbf{y})$ in (2) makes Langevin Monte Carlo applicable in Bayesian problems.

To fully specify a Langevin Monte Carlo algorithm, the location $\boldsymbol{\mu}(\boldsymbol{\theta}, G, \epsilon)$ and covariance $\Sigma(G, \epsilon)$ of normal proposal (1) need to be defined. In what follows, variations of geometric Langevin Monte Carlo methods are distinguished by their respective proposal location and covariance.

2.1 Metropolis-Adjusted Langevin Algorithm

If G is kept constant, that is if G is a not a function of the current state $\boldsymbol{\theta}$, then the Metropolis-adjusted Langevin algorithm (MALA) arises as

$$\boldsymbol{\mu}(\boldsymbol{\theta}, G, \epsilon) = \boldsymbol{\theta} + \frac{\epsilon^2}{2} G^{-1} \nabla_\theta \log p(\boldsymbol{\theta}), \tag{3}$$

$$\Sigma(G, \epsilon) = \epsilon^2 G^{-1}. \tag{4}$$

G is known as the precondition matrix, see Roberts and Stramer (2002). It is typically set to be the identity matrix $G = I$, in which case MALA is defined in its conventional form

$$\boldsymbol{\mu}(\boldsymbol{\theta}, \epsilon) = \boldsymbol{\theta} + \frac{\epsilon^2}{2} \nabla_\theta \log p(\boldsymbol{\theta}),\ \Sigma(\epsilon) = \epsilon^2 I. \tag{5}$$

According to the theoretical analysis of Roberts and Rosenthal (1998), the optimal scaling ϵ has been found to be the value of ϵ which yields a limiting acceptance rate of 57.4% in high-dimensional parametric spaces (as $n_\theta \to \infty$).

2.2 Manifold Langevin Monte Carlo Algorithms

MALA uses the gradient flow $\nabla_\theta \log p(\boldsymbol{\theta})$ to make proposals effectively. As explained by Girolami and Calderhead (2011), it is possible to incorporate further geometric structure in the form of a position-dependent metric $G(\boldsymbol{\theta})$. This way, the Langevin diffusion is defined on a Riemann manifold endowed by the metric $G(\boldsymbol{\theta})$, and the corresponding Levi-Civita connection specified by the Christoffel symbols of second-kind $\Gamma_{ij}^k(\boldsymbol{\theta})$. The resulting manifold Metropolis-adjusted Langevin algorithm (MMALA) draws candidate states from a normal proposal with location and covariance given by

$$\boldsymbol{\mu}_{(k)}(\boldsymbol{\theta}, G(\boldsymbol{\theta}), \epsilon) = \boldsymbol{\theta} + \frac{\epsilon^2}{2} (G^{-1}(\boldsymbol{\theta}) \nabla_\theta \log p(\boldsymbol{\theta}))_{(k)} - \epsilon^2 \sum_{i,j=1}^{n_\theta} G_{ij}^{-1}(\boldsymbol{\theta}) \Gamma_{ij}^k(\boldsymbol{\theta}), \tag{6}$$

$$\Sigma(G(\boldsymbol{\theta}), \epsilon) = \epsilon^2 G^{-1}(\boldsymbol{\theta}), \tag{7}$$

where $\boldsymbol{\mu}_{(k)}$ refers to the k-th coordinate of proposal location $\boldsymbol{\mu}(\boldsymbol{\theta}, G(\boldsymbol{\theta}), \epsilon)$.

If a manfiold of constant curvature is assumed, then the Christoffel symbols vanish, so the proposal location of MMALA simplifies to

$$\boldsymbol{\mu}(\boldsymbol{\theta}, G(\boldsymbol{\theta}), \epsilon) = \boldsymbol{\theta} + \frac{\epsilon^2}{2} G^{-1}(\boldsymbol{\theta}) \nabla_\theta \log p(\boldsymbol{\theta}). \tag{8}$$

The method associated with location (8) is known as simplified Metropolis-adjusted Langeving algorithm (SMMALA). As seen from the location and covariance of their proposal mechanisms, MALA is a special case of SMMALA, which is in turn a special case of MMALA.

The optimal stepsize ϵ for MMALA and SMMALA is empirically suggested by Girolami and Calderhead (2011) to be set so as to obtain an acceptance rate of around 70%; this choice has not been analyzed yet from a theoretical standpoint analogously to the choice of scaling for MALA by Roberts and Rosenthal (1998).

3. Partial Manifold Langevin Metric Updates

3.1 Motivation

To dissect the incurring computational cost of geometric MCMC methods, it suffices to consider their three common steps that dominate computation. In brief, these three steps include sampling from the proposal, evaluating the proposal density, and for some geometric schemes evaluating the derivatives of the metric tensor.

Ordinarily, the proposal used in Langevin Monte Carlo is set to be a multivariate normal distribution $q(\boldsymbol{\theta}^\star|\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta}^\star|\boldsymbol{\mu}(\boldsymbol{\theta}, G, \epsilon), \epsilon^2 G^{-1}(\boldsymbol{\theta}))$, as seen in (1), (4) and (7). To propose a state $\boldsymbol{\theta}^\star$, the proposal is typically simulated in the Cholesky approach by letting

$$\boldsymbol{\theta}^\star = \boldsymbol{\mu}(\boldsymbol{\theta}, G, \epsilon) + \epsilon \left(\sqrt{G^{-1}(\boldsymbol{\theta})}\right)' \tau \tag{9}$$

where $\sqrt{G^{-1}(\boldsymbol{\theta})}$ satisfies the Cholesky decomposition $\left(\sqrt{G^{-1}(\boldsymbol{\theta})}\right)' \sqrt{G^{-1}(\boldsymbol{\theta})} = G^{-1}(\boldsymbol{\theta})$ and $\tau \sim \mathcal{N}(\mathbf{0}, I)$, Chib and Greenberg (1995). So, sampling from the proposal has a complexity of $\mathcal{O}(2n_\theta^3)$, since it requires the inversion of metric $G(\boldsymbol{\theta})$ and the Cholesky decomposition of $G^{-1}(\boldsymbol{\theta})$, each of which are $\mathcal{O}(n_\theta^3)$ operations. MALA, as opposed to SMMALA and MMALA, relies on a constant preconditioning matrix G, therefore G^{-1} and $\sqrt{G^{-1}}$ are evaluated once and cached at the beginning of the simulation avoiding the $\mathcal{O}(n_\theta^3)$ cost.

To calculate the Metropolis-Hastings acceptance ratio (2) appearing in SMMALA and MMALA, it is required to evaluate the normal proposal twice. As it has become apparent, a proposal evaluation has a complexity of $\mathcal{O}(n_\theta^3)$ due to the proposal covariance $\epsilon^2 G^{-1}(\boldsymbol{\theta})$ entailing the inversion of $G(\boldsymbol{\theta})$. Once again, MALA avoids the $\mathcal{O}(n_\theta^3)$ penalty at proposal evaluations as the position-independent inverse G^{-1} is initially computed and is then kept fixed.

MMALA and RMHMC, as introduced by Girolami and Calderhead (2011), additionally require to compute the Christoffel symbols $\Gamma_{ij}^k(\boldsymbol{\theta})$, which involve the partial derivatives $\partial G(\boldsymbol{\theta})/\partial\theta_k,\ k \in \{1, 2, \ldots, n_\theta\}$, of metric $G(\boldsymbol{\theta})$. If $G(\boldsymbol{\theta})$ is chosen to be the observed Fisher information matrix, $\partial G(\boldsymbol{\theta})/\partial\theta_k$ represent the partial derivatives of the negative Hessian, that is they are equal to minus the third-order partial derivatives of the log-target density $\log p(\boldsymbol{\theta})$, thereby the computation of $\partial G(\boldsymbol{\theta})/\partial\theta_k$ is an operation of order $\mathcal{O}(n_\theta^3)$. Since the Christoffel symbols vanish in the case of SMMALA, the cost of $\mathcal{O}(n_\theta^3)$ for the computation of $\partial G(\boldsymbol{\theta})/\partial\theta_k$ is not inflicted on SMMALA.

To sum up, MMALA necessitates computing the two matrix inverses $G^{-1}(\boldsymbol{\theta})$ and $G^{-1}(\boldsymbol{\theta}^\star)$ in order to evaluate the proposals involved in the acceptance probability (2), one Cholesky decomposition $\sqrt{G^{-1}(\boldsymbol{\theta})}$ to sample from the proposal, and the third-order derivatives $\partial G(\boldsymbol{\theta})/\partial\theta_k$ of $G(\boldsymbol{\theta})$ for learning the curvature of the associated manifold. Hence, MMALA has a complexity of $\mathcal{O}(4n_\theta^3)$. SMMALA poses the same computational requirements apart from setting $\partial G(\boldsymbol{\theta})/\partial\theta_k$ to zero, so the complexity of SMMALA becomes $\mathcal{O}(3n_\theta^3)$.

Owing to the works of Davie and Stothers (2013), Williams (2011) and Gall (2014), optimized versions of the Coppersmith-Winograd algorithm allow to perform matrix multiplication and therefore matrix inversion in $\mathcal{O}(n_\theta^{2.373})$ time. Consequently, optimized implementations of MMALA and SMMALA can run in $\mathcal{O}(2n_\theta^3 + 2n_\theta^{2.373})$ and $\mathcal{O}(n_\theta^3 + 2n_\theta^{2.373})$ time, respectively.

A detailed account of the time complexity of SMMALA has been given to highlight the main sources of computational load involved in geometric updates. In what follows, scaling factors and constants will be suppressed in big-$\mathcal{O}$ notation following common practice.

Since the preconditioning matrix G, its inverse G^{-1} and the Cholesky decomposition $\sqrt{G^{-1}}$ are cached upon initialization for MALA, the quadratic form $(\boldsymbol{\theta}^\star - \boldsymbol{\theta}^\star)' G^{-1} (\boldsymbol{\theta}^\star - \boldsymbol{\theta}^\star)$ in the normal proposal density dictates a time complexity of $\mathcal{O}(n_\theta^2)$ for MALA. If G is set to be the identity matrix, then $(\boldsymbol{\theta}^\star - \boldsymbol{\theta}^\star)' G^{-1} (\boldsymbol{\theta}^\star - \boldsymbol{\theta}^\star)$ reduces to the inner product

$\langle \boldsymbol{\theta}^\star - \boldsymbol{\theta}^\star, \boldsymbol{\theta}^\star - \boldsymbol{\theta}^\star \rangle$. This inner product and the gradient $\nabla_\theta \log p(\boldsymbol{\theta})$ of the log-target have a complexity of $\mathcal{O}(n_\theta)$ each. So, if $G = I$, the complexity of MALA is of order $\mathcal{O}(n_\theta)$.

An implicit assumption made for all the complexity bounds of this section is that the log-target evaluations do not exceed the complexity of the associated samplers. Of course there are models for which such assumption is not plausible. For instance, if the underlying model consists of a system of non-linear ordinary differential equations (ODEs) and each log-target calculation requires solving the ODE system numerically, then the log-target evaluations might dominate the cost of the MCMC simulation depending on the size of the parameter and state space and on the complexity of the numerical ODE solver.

Combining the best of two worlds, an effective and efficient MCMC algorithm would exploit geometric information in time as close as possible to the linear time of MALA. One potential approach towards this end would be to approximate the expensive components of geometric computation, such as $G^{-1}(\boldsymbol{\theta})$ and $\sqrt{G^{-1}(\boldsymbol{\theta})}$, in the spirit of Şimşekli et al. (2016). An alternative route, taken by the present paper, is to design a hybrid sampling method that switches between expensive geometrically-informed and cheaper non-geometric proposal mechanisms. The latter idea is motivated by the objective to exploit exact geometric updates whenever these occur, therefore not rendering each Monte Carlo iteration approximate.

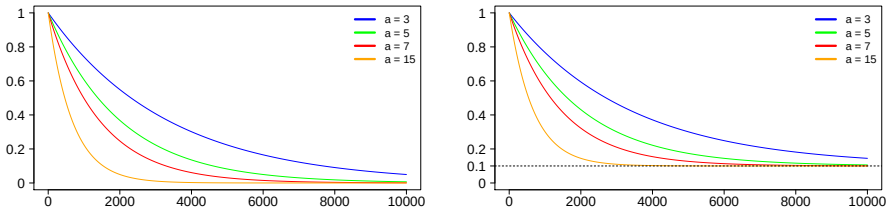
3.2 Partial Manifold Langevin Metric Updates in MALA

Sections 2.1 and 2.2 provide a unified formulation of MALA and SMMALA. More specifically, both samplers constitute variations of a Metropolis-Hastings algorithm with a normal proposal given by (1) and acceptance probability defined by (2). The MALA and SMMALA normal proposals differ in their locations and covariances, which are specified by (3)-(4) and (8)-(7) respectively. From an algorithmic perspective, the natural way to switch between MALA and SMMALA updates is to variably rely on the locations and covariances of their associated candidate-generating kernels.

Both MALA and SMMALA learn from the local geometry of the underlying manifold. MALA resorts to gradient flow computations, while SMMALA further enriches its geometric information by incorporating the position-specific metric $G(\boldsymbol{\theta})$ to the updating mechanism. So, strictly speaking, a hybrid algorithm that switches between MALA and SMMALA proposals combines two geometric MCMC methods with varying levels of geometric information and of computational cost. To keep the resulting complexity as low as possible, the bulk of transitions should be driven by MALA, allowing intervening SMMALA updates to occur just frequently enough to boost sampling effectiveness.

A first question is how to pick from the MALA and SMMALA proposals at each MCMC iteration. One possibility is to make a random choice via a Bernoulli trial. At the i-th iteration, draw a sample $b(i) \sim \text{Bernoulli}(p(i))$ with probability $p(i)$ of taking a SMMALA step. If $b(i) = 0$ then propose a new state $\boldsymbol{\theta}^\star$ sampled from the MALA proposal, whereas if $b(i) = 1$ then propose $\boldsymbol{\theta}^\star$ drawn from the SMMALA proposal.

Secondly, there needs to be a way of selecting the probabilities $p(i)$ as a function of the current iteration $i \in \{1, 2, \cdots, n_m\}$, where n_m is the total number of MCMC steps. In other words, a schedule is needed for the probabilities $p(i)$ of choosing between MALA and



(a) $b = 0$ (b) $b = 0.1$

Figure 1: Exponentially decaying schedule of probabilities $p(i)$ of making SMMALA proposals as a function of i-th MCMC iteration. The tuning factor b sets the asymptotic probability of selecting the SMMALA proposal mechanism over its MALA counterpart. On the other hand, the tuning factor a determines the rate of decay of $p(i)$ with larger a corresponding to more rapid decay.

SMMALA updates. One possible choice of $p(i)$ is an exponentially decaying schedule

$$p(i) = \exp\left(-a(i-1)/n_m\right), \tag{10}$$

where a denotes a constant tuning factor that determines the decay rate of $p(i)$. Schedule (10) makes SMMALA updates more probable in early transient iterations of the chain, while it makes MALA updates increasingly more likely in late stationary iterations. Hence, as the chain converges, less geometric information is acquired to reduce the computational load. Figure 1a displays the schedule of Bernoulli probabilities for varying decaying rates.

Subject to the model under consideration, it might be desirable to maintain a certain probability b of making SMMALA proposals in late MCMC iterations instead of eliminating this probability. Schedule (10) can be modified as

$$p(i) = (1-b)\exp\left(-a(i-1)/n_m\right) + b, \tag{11}$$

where the additional tuning factor b sets the asymptotic probability of using the SMMALA proposal mechanism in late MCMC iterations. Obviously (10) arises as a special case of (11) by setting $b = 0$ in the latter equation. Figure 1b visualizes schedule (11) for varying decaying rates and for $b = 0.1$, which corresponds to a 10% probability of picking SMMALA proposals as the simulation progresses towards its end.

A third and last question to be addressed in order to fully construct the hybrid sampler is how to set the preconditioning matrix for MALA proposals. One way to proceed is to use the most recent metric $G(\boldsymbol{\theta}^0)$ computed at the latest state $\boldsymbol{\theta}^0$ obtained via a SMMALA update. This is where the computational gain comes from, since $G(\boldsymbol{\theta}^0)$, $G^{-1}(\boldsymbol{\theta}^0)$ and $\sqrt{G^{-1}(\boldsymbol{\theta}^0)}$ are cached and reused for the intervening MALA steps between successive SMMALA updates. An alternative option is to utlilize an identity preconditioning matrix $G = I$ corresponding to MALA proposal densities of the form $\mathcal{N}(\boldsymbol{\theta} + \frac{\epsilon^2}{2}\nabla_\theta \log p(\boldsymbol{\theta}), \epsilon^2 I)$, thus adhering to the most commonly used version of MALA.

Algorithm 1 ALSMMALA

$\boldsymbol{\theta}^{\circ} = \boldsymbol{\theta}$

for $k = 1$ to n_m **do**
  Sample $b \sim \text{Bernoulli}(p(i))$

  Sample $u \sim \mathcal{U}(0, 1)$

  if $b = 0$ **then** ▷ Use MALA proposal
    Sample $\boldsymbol{\theta}^{*} \sim \mathcal{N}(\boldsymbol{\mu}(\boldsymbol{\theta}, G(\boldsymbol{\theta}^{\circ}), \epsilon), \epsilon^2 G^{-1}(\boldsymbol{\theta}^{\circ}))$

    if $u < \dfrac{p(\boldsymbol{\theta}^{*})\mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\mu}(\boldsymbol{\theta}^{*}, G(\boldsymbol{\theta}^{\circ}), \epsilon), \epsilon^2 G^{-1}(\boldsymbol{\theta}^{\circ}))}{p(\boldsymbol{\theta})\mathcal{N}(\boldsymbol{\theta}^{*}|\boldsymbol{\mu}(\boldsymbol{\theta}, G(\boldsymbol{\theta}^{\circ}), \epsilon), \epsilon^2 G^{-1}(\boldsymbol{\theta}^{\circ}))}$ **then**
      $\boldsymbol{\theta} = \boldsymbol{\theta}^{*}$
    end if
  else if $b = 1$ **then** ▷ Use SMMALA proposal
    Sample $\boldsymbol{\theta}^{*} \sim \mathcal{N}(\boldsymbol{\mu}(\boldsymbol{\theta}, G(\boldsymbol{\theta}), \epsilon), \epsilon^2 G^{-1}(\boldsymbol{\theta}))$

    if $u < \dfrac{p(\boldsymbol{\theta}^{*})\mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\mu}(\boldsymbol{\theta}^{*}, G(\boldsymbol{\theta}^{*}), \epsilon), \epsilon^2 G^{-1}(\boldsymbol{\theta}^{*}))}{p(\boldsymbol{\theta})\mathcal{N}(\boldsymbol{\theta}^{*}|\boldsymbol{\mu}(\boldsymbol{\theta}, G(\boldsymbol{\theta}), \epsilon), \epsilon^2 G^{-1}(\boldsymbol{\theta}))}$ **then**
      $\boldsymbol{\theta} = \boldsymbol{\theta}^{*}$
    end if

    $\boldsymbol{\theta}^{\circ} = \boldsymbol{\theta}$
  end if
end for

The resulting algorithm is abbreviated as ALSMMALA. The AL prefix implies that the bulk of states are drawn from a MALA proposal mechanism and the SMMALA suffix indicates that the local geometry is exploited via more expensive SMMALA updates. Algorithm 1 provides pseudocode to summarize ALSMMALA for some schedule $p(i)$.

3.3 Partial Manifold Langevin Metric Updates in Adaptive Metropolis

The focal idea of this paper extends beyond ALSMMALA. In principle, it is suggested to allow the proposal mechanism to vary between MCMC iterations. For example, most states can be sampled from a cheaper candidate-generating kernel, whereas fewer states can be drawn from a more costly proposal that uses local geometric information.

The candidate-generating kernel that contributes the majority of states to a hybrid sampler with a dual proposal mechanism impacts the statistical properties of the sampler. For instance, Roberts and Tweedie (1996) and Livingstone and Girolami (2014) have shown that if a target density has tails heavier than exponential or lighter than Gaussian, then a MALA proposal kernel does not yield a geometrically ergodic Markov chain. For targets with such tail behaviour, ALSMMALA is not anticipated to admit geometric ergodicity either.

Depending on the target density, different proposal mechanisms might need to be combined to attain a hybrid sampler with the desired statistical properties and sampling efficiency. In order to define a viable complement to ALSMMALA and to further explore how to combine heterogeneous proposal mechanisms under a single MCMC algorithm, one more hybrid sampler with partial geometric updates will be constructed.

The new algorithm will sample most of its states from a proposal kernel inspired by the adaptive Metropolis (AM) kernel of Haario et al. (2001), while it will draw fewer states from the SMMALA proposal kernel. Given its candidate-generating constituents, the new algorithm will be called AMSMMALA. The examples of Section 4 will demonstrate the relative performance of ALSMMALA and AMSMMALA for different target densities.

Initially, a concise account of the adaptive Metropolis algorithm of Haario et al. (2001) is given. Consider a hitherto simulated chain $(\boldsymbol{\theta}_0, \boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_k)$, denoted by $\boldsymbol{\theta}_{0:k}$. It is noted that $\boldsymbol{\theta}_i \in \mathbb{R}^{n_\theta}$ refers to the i-th n_θ-dimensional state of the chain, to be distinguished from the parenthesized subscript $\boldsymbol{\theta}_{(i)} \in \mathbb{R}$, which represents the i-th scalar coordinate of some state $\boldsymbol{\theta} \in \mathbb{R}^{n_\theta}$. A candidate state $\boldsymbol{\theta}^\star$ is sampled from the normal density $\mathcal{N}(\boldsymbol{\theta}_k, \hat{C}(\boldsymbol{\theta}_k|\boldsymbol{\theta}_{0:k-1}))$. This proposal density is commonly centred on the last state $\boldsymbol{\theta}_k$ and its covariance matrix $\mathrm{Cov}(\boldsymbol{\theta}_k)$ is estimated adaptively by

$$\hat{C}(\boldsymbol{\theta}_k|\boldsymbol{\theta}_{0:k-1}) = s_{n_\theta}\widehat{\mathrm{Cov}}(\boldsymbol{\theta}_k|\boldsymbol{\theta}_{0:k-1}) + s_{n_\theta}\lambda I, \tag{12}$$

where $\widehat{\mathrm{Cov}}(\boldsymbol{\theta}_k|\boldsymbol{\theta}_{0:k-1})$ is the empirical covariance matrix

$$\widehat{\mathrm{Cov}}(\boldsymbol{\theta}_k|\boldsymbol{\theta}_{0:k-1}) = \frac{1}{k-1}\left(\sum_{i=0}^{k-1}\boldsymbol{\theta}_i\boldsymbol{\theta}_i^{'} - k\bar{\boldsymbol{\theta}}_{k-1}\bar{\boldsymbol{\theta}}_{k-1}^{'}\right). \tag{13}$$

$\bar{\boldsymbol{\theta}}_{k-1}$ denotes the sample mean of $\boldsymbol{\theta}_{0:k-1}$. The tuning parameter s_{n_θ} depends only on the dimension n_θ of the parameter space and is used for scaling the empirical covariance. Furthermore, $\lambda > 0$ is another tuning constant, which is chosen to be very small and is needed for preventing the estimator $\hat{C}(\boldsymbol{\theta}_k|\boldsymbol{\theta}_{0:k-1})$ from becoming degenerate for larger k. The candidate state $\boldsymbol{\theta}^\star$ is accepted with probability

$$a(\boldsymbol{\theta}_k, \boldsymbol{\theta}^\star) = \min\left\{\frac{p(\boldsymbol{\theta}^\star)}{p(\boldsymbol{\theta}_k)}, 1\right\}. \tag{14}$$

Since the proposal kernel $\mathcal{N}(\boldsymbol{\theta}^\star|\boldsymbol{\theta}_k, \hat{C}(\boldsymbol{\theta}_k|\boldsymbol{\theta}_{0:k-1}))$ is symmetric, it does not appear in the acceptance ratio (14).

To enable switching betweeen AM and SMMALA transitions, the covariance matrices of the two involved proposals must be brought together under a common framework. If a SMMALA update is attempted at the k-th MCMC iteration, the local covariance at $\boldsymbol{\theta}_k$ is evaluated exactly as $\mathrm{Cov}(\boldsymbol{\theta}_k) = \epsilon^2 G^{-1}(\boldsymbol{\theta}_k)$. If an adjusted Metropolis update is tried at the k-th iteration, the metric $G(\boldsymbol{\theta}_k)$ is not computed, therefore an estimator $\widehat{G^{-1}}(\boldsymbol{\theta}_k|\boldsymbol{\theta}_{0:k-1})$ of $G^{-1}(\boldsymbol{\theta}_k)$ must be obtained from the empirical covariance of AM. Setting the diffusion drift step ϵ temporarily aside, it becomes clear that $G^{-1}(\boldsymbol{\theta}_k)$ will either be evaluated exactly by SMMALA or it will be estimated by the empirical covariance $\widehat{G^{-1}}(\boldsymbol{\theta}_k|\boldsymbol{\theta}_{0:k-1}) = \widehat{\mathrm{Cov}}(\boldsymbol{\theta}_k|\boldsymbol{\theta}_{0:k-1})$, of the AM proposal as given by (13).

Algorithm 2 AMSMMALA

for $k = 1$ to n_m **do**
  if $k \equiv 0$ (mod (a)) **then**
    $b = 1$
  else
    $b = 0$
  end if

  Sample $u \sim \mathcal{U}(0, 1)$

  if $b = 0$ **then** ▷ Use adaptive Metropolis proposal
    Sample $\boldsymbol{\theta}^* \sim \mathcal{N}(\boldsymbol{\theta}, \epsilon^2 \widehat{G^{-1}}(\boldsymbol{\theta}|\boldsymbol{\theta}_{0:k-1}))$

    if $u < \dfrac{p(\boldsymbol{\theta}^*)}{p(\boldsymbol{\theta})}$ **then**
      $\boldsymbol{\theta} = \boldsymbol{\theta}^*$
    end if
  else if $b = 1$ **then** ▷ Use SMMALA proposal
    Sample $\boldsymbol{\theta}^* \sim \mathcal{N}(\boldsymbol{\mu}(\boldsymbol{\theta}, G(\boldsymbol{\theta}), \epsilon), \epsilon^2 G^{-1}(\boldsymbol{\theta}))$

    if $u < \dfrac{p(\boldsymbol{\theta}^*)\mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\mu}(\boldsymbol{\theta}^*, G(\boldsymbol{\theta}^*), \epsilon), \epsilon^2 G^{-1}(\boldsymbol{\theta}^*))}{p(\boldsymbol{\theta})\mathcal{N}(\boldsymbol{\theta}^*|\boldsymbol{\mu}(\boldsymbol{\theta}, G(\boldsymbol{\theta}), \epsilon), \epsilon^2 G^{-1}(\boldsymbol{\theta}))}$ **then**
      $\boldsymbol{\theta} = \boldsymbol{\theta}^*$
    end if
  end if
end for

As long as SMMALA steps are embedded throughout the AMSMMALA simulation, the AM estimator $\widehat{G^{-1}}(\boldsymbol{\theta}_k|\boldsymbol{\theta}_{0:k-1})$ is corrected by the exact inverse metric evaluation at the most recently accepted SMMALA update. Put another way, recurring geometric updates prevent the empirical covariance of AM from becoming degenerate. Hence, the $s_{n_\theta}\lambda I$ term in (12) is omitted by setting $\lambda = 0$. Moreover, the tuning parameter s_{n_θ} of AM is set to $s_{n_\theta} = \epsilon^2$ in (12) so that $\widehat{G^{-1}}(\boldsymbol{\theta}_k|\boldsymbol{\theta}_{0:k-1})$ and $G^{-1}(\boldsymbol{\theta}_k)$ are coherently scaled by the same factor in AM and SMMALA updates respectively. Setting $\lambda = 0$ and $s_{n_\theta} = \epsilon^2$ in (12) defines the empirical covariance estimator $\hat{C}(\boldsymbol{\theta}_k|\boldsymbol{\theta}_{0:k-1}) = \epsilon^2\widehat{G^{-1}}(\boldsymbol{\theta}_k|\boldsymbol{\theta}_{0:k-1})$ for the AM proposal density $\mathcal{N}(\boldsymbol{\theta}_k, \epsilon^2\widehat{G^{-1}}(\boldsymbol{\theta}_k|\boldsymbol{\theta}_{0:k-1}))$ at the k-th step of AMSMMALA, where $\widehat{G^{-1}}(\boldsymbol{\theta}_k|\boldsymbol{\theta}_{0:k-1})$ is given by (13).

To ensure that the adaptive Metropolis covariance estimator receives exact geometric updates regularly, a mod (a) scheduler can be employed. If the k-th MCMC iteration is a multiple of a tuning constant a, then a SMMALA update is attempted, else an AM proposal is made. Algorithm 2 summarizes AMSMMALA with the help of pseudocode.

It follows from (13) that the empirical inverse metric of AMSMMALA used at the AM updates calculates recursively as

$$(k-1)\widehat{G^{-1}}(\boldsymbol{\theta}_k) = (k-2)\widehat{G^{-1}}(\boldsymbol{\theta}_{k-1}) + \boldsymbol{\theta}_{k-1}\boldsymbol{\theta}_{k-1}^{'} - k\bar{\boldsymbol{\theta}}_{k-1}\bar{\boldsymbol{\theta}}_{k-1}^{'} + (k-1)\bar{\boldsymbol{\theta}}_{k-2}\bar{\boldsymbol{\theta}}_{k-2}^{'}, \quad (15)$$

where $\widehat{G^{-1}}(\boldsymbol{\theta}_k)$ is a shorthand for $\widehat{G^{-1}}(\boldsymbol{\theta}_k|\boldsymbol{\theta}_{0:k-1})$. The sample means in (15) are also calculable recursively according to $k\bar{\boldsymbol{\theta}}_k = (k-1)\bar{\boldsymbol{\theta}}_{k-1} + \boldsymbol{\theta}_k$. Apparently, (15) computes the empirical covariance of AMSMMALA faster than (13).

The AM steps of AMSMMALA waive the cost of proposal evaluations thanks to the acceptance probability (14) and do not require computing log-target derivatives. However, sampling a candidate state from the AM proposal kernel creates a complexity bottleneck of $\mathcal{O}(n_\theta^3)$ due to the associated Cholesky decomposition of the empirical covariance. The recursive formula (15) comes to rescue, as it allows to replace the Cholesky factorization by two rank one updates and one rank one downdate, thus reducing the complexity of AM updates from $\mathcal{O}(n_\theta^3)$ to $\mathcal{O}(3n_\theta^2)$. Gill et al. (1974) and Seeger (2004) elaborate on low rank updates for Cholesky decompositions.

AMSMMALA yields two more advantages over AM apart from exploiting local geometric information. An initial covariance matrix $\hat{C}_0 = \epsilon^2\widehat{G^{-1}}(\boldsymbol{\theta}_0)$ is needed for starting up the recursion in (15). Haario et al. (2001) propose to choose $\hat{C}_0$ on the basis of prior knowledge. If prior elicitation for $\hat{C}_0$ turns to be non-trivial, AMSMMALA gives the option to initialize $C_0 = \epsilon^2 G^{-1}(\boldsymbol{\theta}_0)$ by a SMMALA update. Additionally, the recurring corrections of the empirical covariance via exact SMMALA updates make AMSMMALA available to target densities of non-bounded support, extending this way the range of applicability of the original AM algorithm by Haario et al. (2001).

3.4 Choice of Update Schedule

The two proposal mechanisms along with the schedule for picking between them comprise the three main building blocks of any resulting hybrid sampler. ALSMMALA and AMSMMALA have manifested ways of combining cheap with more expensive geometric updates to construct a new MCMC algorithm. The underlying proposals set some limitations in the choice of schedule. AMSMMALA for example needs to receive exact metric updates throughout the course of simulation to correct the empirical covariance used in adaptive Metropolis steps. Despite any inherent limitations posed by the proposal mechanisms, there is room for experimentation when choosing a schedule.

For instance, a geometric distribution Geometric(p) could be used for regulating the occurrence of metric updates in AMSMMALA, where p is a tuning parameter giving the probability of making a SMMALA proposal. Setting $p = 1/(1+a)$ yields an expected number of $(1-p)/p = a$ AM steps before a new SMMALA proposal is made. This way, the Geometric($1/(1+a)$) schedule is a stochastic analogue to the deterministic $\bmod(a)$ schedule of Algorithm 2.

Other samplers with dual proposal mechanism, such as ALSMMALA, do not need regular geometric steps, thence it makes sense to exploit local geometric information mostly in the early transient phases of the simulated chain and to reduce computational load via a cheaper proposal in late stationary phases. A Bernoulli(p(i)) distribution comes in handy, leaving space for experimentation with the choice of schedule for $p(i)$. The probability $p(i)$ plays a role analogous to temperature in the cooling schedule of simulated annealing. This analogy brings up a prolific body of literature to open up possibilities for scheduling the cooling of probability $p(i)$ of geometric updates.

Cooling	$p(i)$
Exponential	$(1-b)\exp(-a(i-1)/n_m)+b$
Linear	$\dfrac{1-b}{1+a(i-1)/n_m}+b$
Quadratic	$\dfrac{1-b}{1+a((i-1)/n_m)^2}+b$
Logarithmic	$\dfrac{(1-b)}{1+a\log(1+(i-1)/n_m)}+b$

Table 1: Different types of cooling schedule for probability $p(i),\ i \in \{1,2,\ldots,n_m\}$, of making a geometric proposal as a function of the i-th MCMC iteration. Such cooling for $p(i)$ is plausible if the dual proposal scheme does not require regular geometric updates throughout the simulation.

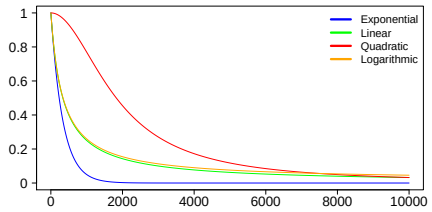


Figure 2: Visualization of four cooling schedule types of Table 1 for probability $p(i)$ of making a geometric proposal as a function of the i-th MCMC iteration. The displayed schedules have been generated using $a = 30$ and $b = 0$.

Table 1 suggests four cooling schedules for $p(i)$, which have been acquired empirically and are variants of temperature schedules previously discussed by Hajek (1987) and Nourani and Andresen (1998). Figure 2 displays instances of the four schedules of Table 1.

3.5 Expected Complexity

The concept of complexity carries three distinct meanings in the context of MCMC. Firstly, MCMC methods need to be tuned so as to achieve a balance between proposing large enough jumps and ensuring that a reasonable proportion of jumps are accepted. By way of illustration, MALA attains its optimal acceptance rate of 0.574 as $n_\theta \to \infty$ by setting its drift step ϵ to be in the vicinity of $n_\theta^{-1/3}$. Because of this, it is said that the algorithmic efficiency of MALA scales $\mathcal{O}(n_\theta^{-1/3})$ as the number n_θ of parameters increases.

Secondly, the quality of MCMC algorithms is assessed by estimating the effective sample size (ESS) of their simulated chains. The ESS of a chain is interpreted as the number

of samples in the chain bearing the same amount of variance as the one found in n_m independent samples.

A third criterion for assessing MCMC methods is their computational time. This criterion corresponds to the ordinary concept of algorithmic complexity as it entails a count of numerical operations performed by an MCMC algorithm. To give an example, the computational complexity of MALA is of order $\mathcal{O}(n_\theta^2)$ or $\mathcal{O}(n_\theta)$ depending on the choice of preconditioning matrix, as explained in Section 3.1.

Of these three indicators of algorithmic complexity, ESS and computational runtime are the ones typically used for understanding the range of applicability of MCMC methods. To get a single-number summary, the ratio of ESS over runtime is usually employed. In the present paper, the computational time of ALSMMALA and AMSMMALA are derived theoretically, while both samplers are empirically assessed in Section 4 via their ESS and CPU runtime.

Proposition 1 *Let A_1 and A_2 be two MCMC algorithms whose proposal mechanisms have computational complexities c_1 and c_2, respectively. A composite MCMC algorithm A conducts a Bernoulli trial $B(i) \sim Bernoulli(p(i))$, $i \in \{1, 2, \ldots, n_m\}$, to take its i-th step using the proposal kernel of algorithm A_1 with probability $p(i)$, or using the proposal mechanism of A_2 with probability $1 - p(i)$. The mean complexity of A is given by*

$$\mathcal{O}\left(\frac{\sum_{i=1}^{n_m} p(i)}{n_m} c_1 + \frac{n_m - \sum_{i=1}^{n_m} p(i)}{n_m} c_2\right). \tag{16}$$

Proof The expected number of invocations of the proposal kernel associated with algorithm A_1 equals

$$E\left(\sum_{i=1}^{n_m} B(i)\right) = \sum_{i=1}^{n_m} E(B(i)) = \sum_{i=1}^{n_m} p(i),$$

whence the conclusion follows directly. ■

Proposition 2 *If an exponentially decaying schedule $p(i)$, as described by* (10)*, is used for regulating the choice of proposal kernel at each step of algorithm A, then the expected complexity of A becomes*

$$\mathcal{O}\left(\frac{1 - \exp(-a)}{n_m(1 - \exp(-a/n_m))} c_1 + \left(1 - \frac{1 - \exp(-a)}{n_m(1 - \exp(-a/n_m))}\right) c_2\right). \tag{17}$$

Proof Schedule (10) yields the geometric series sum

$$\sum_{i=1}^{n_m} p(i) = \sum_{i=1}^{n_m} \exp(-a(i-1)/n_m) = \frac{1 - \exp(-a)}{1 - \exp(-a/n_m)}. \tag{18}$$

Plugging (18) into (16) gives (17). ■

Lemma 3 *For sufficiently large number n_m of MCMC iterations ($n_m \to \infty$), the mean complexity of an algorithm A with exponentially decaying schedule* (10) *is.*

$$\mathcal{O}\left(\frac{1-\exp(-a)}{a}c_1 + \left(1 - \frac{1-\exp(-a)}{a}\right)c_2\right). \tag{19}$$

Proof It suffices to notice that

$$\lim_{n_m \to \infty} \frac{1-\exp(-a)}{n_m(1-\exp(-a/n_m))} = \frac{1-\exp(-a)}{a}. \tag{20}$$

■

Proposition 4 *The mean complexity of an algorithm A with a* $\mod(a)$ *schedule $p(i)$ is*

$$\mathcal{O}\left(\frac{1}{a}c_1 + \frac{a-1}{a}c_2\right). \tag{21}$$

Proof The proposal mechanism of algorithm c_1 is called n_m/a times out of n_m iterations. Hence, the proportion of proposals made via c_1 is $1/a$, which completes the proof. ■

Lemma 5 *For a large number n_m of iterations ($n_m \to \infty$), the expected complexity of ALSMMALA, as presented in Algorithm 1 and equipped with the exponentially decaying schedule defined by* (10)*, is given by*

$$\mathcal{O}\left(\frac{1-\exp(-a)}{a}n_\theta^3 + \left(1 - \frac{1-\exp(-a)}{a}\right)n_\theta^2\right). \tag{22}$$

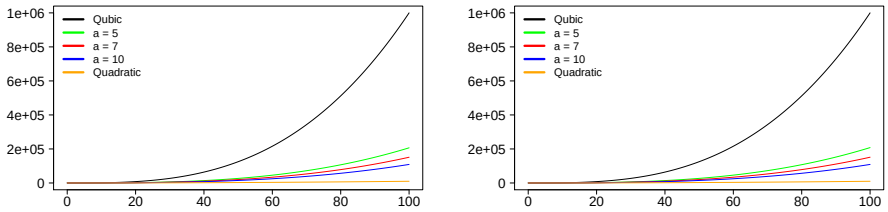
Proof (22) follows from (19) by recalling that the respective complexities of SMMALA and MALA are $\mathcal{O}(n_\theta^3)$ and $\mathcal{O}(n_\theta^2)$. ■

Lemma 6 *The expected complexity of AMSMMALA, as defined in Algorithm 2, is*

$$\mathcal{O}\left(\frac{1}{a}n_\theta^3 + \frac{a-1}{a}n_\theta^2\right). \tag{23}$$

Proof Setting $c1 = n_\theta^3$ and $c_2 = n_\theta^2$ in (21), which correspond to the complexity orders of SMMALA and AM, gives (23). ■

The schedule $p(i)$ for choosing between proposal mechanisms plays a decisive role in the complexity of the resulting sampler as seen from (16). Besides, the decaying rate a of an exponentially decaying schedule (10) and the modulus a of a $\mod(a)$ schedule can be used for adjusting the computational cost according to (19) and (21).



(a) ALSMMALA (b) AMSMMALA

Figure 3: Time complexity of ALSMMALA with an exponential schedule given by (10) and AMSMMALA with a mod (a) schedule as a function of the number n_θ of parameters. Different curves correspond to varying levels of decaying rate a for ALSMMALA and of modulus a for AMSMMALA. The qubic complexity of SMMALA and quadratic complexity of MALA are displayed as baseline cases.

Figure 3 displays the complexity of ALSMMALA with an exponential schedule specified by (10) and AMSMMALA with a mod (a) schedule as a function of the parameter space dimension n_θ. The curves of Figures 3a and 3b correspond to varying levels of decaying rate a for ALSMMALA and of modulus a for AMSMMALA. In both cases, the tuning factor a determines the trade-off between runtime and amount of geometric computation. Larger values of a reduce the runtime due to fewer geometric updates, whereas smaller values of a increase the complexity as they induce more geometric updates.

3.6 Analytically Intractable Geometric Updates

In practice, challenges in the implementation of manifold MCMC algorithms might raise additional computational implications. In particular, two notoriously incurring issues relate to the Cholesky decomposition of metric $G^{-1}(\boldsymbol{\theta})$ and to the calculation of up to third order derivatives of $G(\boldsymbol{\theta})$.

Various contributing factors, including finite-precision floating point arithmetic, can lead to an indefinite proposal covariance matrix $\epsilon^2 G^{-1}(\boldsymbol{\theta})$. This in turn breaks the Cholesky factorization of $\epsilon^2 G^{-1}(\boldsymbol{\theta})$. Betancourt (2013) introduced the so-called SoftAbs metric, which offers a positive definite approximation of the indefinite proposal covariance. The working SoftAbs solution to the Cholesky factorization problem involves an eigen-decomposition, so it comes with an $\mathcal{O}(n_\theta^3)$ cost. Other research avenues pursuit different positive definite approximations of indefinite matrices. The works of Higham (1988) and Higham (2002) for example approximate the nearest symmetric positive semi-definite matrix in the Frobenius norm to an arbitrary real matrix. If future research succeeds in adapting the approach of Higham (1988) in the context of geometric MCMC methods, then a computationally cheaper alternative to the SoftAbs metric could become available.

Non-trivial models can render the analytic derivation of log-target derivatives impossible or impractical. Automatic differentiation (AD), a computationally driven research activity that has evolved since the mid 1950's, helps compute derivatives in a numerically exact way.

Indeed, Griewank (2014) has shown that AD is backward stable in the sense of Wilkinson (1971). Thus, small perturbations of the original function due to machine precision still yield accurate derivatives calculated via AD.

There are different methods of automatic differentiation; reverse mode AD is better suited for functions $f : \mathbb{R}^n \to \mathbb{R}$, in contrast to forward mode AD that is more suitable for functions $f : \mathbb{R} \to \mathbb{R}^n$ (Griewank and Walther (2008)). Consequently, reverse mode AD is utilized for computing derivatives of probability distributions, and finds use in statistical inference. The complexity of reverse mode AD is not worse than that of the respective analytical derivatives of a target density, assuming of course a carefully crafted implementation.

4. Examples

ALSMMALA and AMSMMALA are put to test via three examples to empirically validate these two MCMC algorithms and to assess their computational efficacy. Ten chains are simulated from each sampler of each example. $110,000$ iterations are run for the realization of each chain, of which the first $10,000$ burn-in are discarded, so $n_m = 100,000$ MCMC samples are retained.

The effective sample size (ESS) of each dimension $\boldsymbol{\theta}_{(i)},\ i \in \{1, 2, \ldots, n_\theta\}$, of the parameter vector $\boldsymbol{\theta}$ is computed as

$$\text{ESS} = \frac{n_m \hat{\sigma}^2_{IID}}{\hat{\sigma}^2_{MCMC}}, \tag{24}$$

where $\hat{\sigma}^2_{IID}$ and $\hat{\sigma}^2_{MCMC}$ denote the ordinary and Monte Carlo variance of the associated chain. The Monte Carlo variance $\hat{\sigma}^2_{MCMC}$ is calculated by means of the initial monotone sequence estimator of Geyer (1992). To get more reliable estimates, the ESS of each dimension and the CPU runtime are averaged by taking their respective means across every set of ten chains simulated from a sampler. The computational efficiency of each sampler is defined as the ratio $\min\{\text{ESS}_i : i = 1, 2, \ldots, n_\theta\}$ /time of the smallest ESS $\min\{\text{ESS}_i : i = 1, 2, \ldots, n_\theta\}$ among all n_θ dimensions over the simulation time. Finally, the relative speed-up of one MCMC method over another is set to be the ratio of their computational efficiencies.

A package, called PGUManifoldMC, has been developed to implement ALSMMALA and AMSMMALA using the Julia programming language. PGUManifoldMC is based on Lora, a package for MCMC inference written in Julia by one of the two authors. PGUManifoldMC is publicly available at `https://github.com/scidom/PGUManifoldMC.jl` under an MIT license. The PGUManifoldMC package ships with the three examples of this paper.

4.1 Bayesian Logistic Regression

As a first example, consider a Bayesian logistic regression model consisting of n_d samples and n_θ covariates. The log-likelihood of the model expresses as

$$L(\mathbf{y}, X|\boldsymbol{\theta}) = (X\boldsymbol{\theta})^T \mathbf{y} - \sum_{i=1}^{n_d} \ln\left[1 + \exp(\boldsymbol{\theta}^T \mathbf{x}_{\mathbf{i},})\right], \tag{25}$$

where $y \in \{0, 1\}^{n_d}$ is the binary response variable, X the $n_d \cdot n_\theta$ design matrix, $\mathbf{x}_{\mathbf{i},}$ the i-th row of X and $\boldsymbol{\theta} \in \mathbb{R}^{n_\theta}$ the regression coefficients. A normal prior is assumed for the model

Table 2: Logistic regression - comparison of Langevin sampling methods

Method	AR	ESS				Time	Efficiency	Speedup
		θ_1	θ_2	θ_3	θ_4			
MALA	.60	23077	8039	8892	8562	3.05	2637.47	1.00
SMMALA	.69	15138	15246	15098	12989	6.78	1916.05	0.73
AMSMMALA	.25	9401	9208	9366	8803	3.84	2291.60	0.87
ALSMMALA	.63	31562	31046	30656	26535	4.82	5503.74	2.09

parameters $\boldsymbol{\theta} \sim \pi(\boldsymbol{\theta}) = \mathcal{N}(\mathbf{0}, vI)$ for some positive hyperparameter v. The unnormalized log-target density for Bayesian logistic regression thus takes the form

$$\log\left(p(\boldsymbol{\theta}|\mathbf{y}, X)\right) = L(\mathbf{y}, X|\boldsymbol{\theta}) + \log\left(\pi(\boldsymbol{\theta})\right), \tag{26}$$

where the log-likelihood is given by (25).

Differentiating (25) gives the gradient of the log-likelihood

$$\nabla_{\boldsymbol{\theta}}(L(\mathbf{y}, X|\boldsymbol{\theta})) = X^T\left[\mathbf{y} - \frac{1}{1+\exp\left(-X\boldsymbol{\theta}\right)}\right]. \tag{27}$$

The gradient of the log-prior evaluates as

$$\nabla_{\boldsymbol{\theta}}(\log\left(\pi(\boldsymbol{\theta})\right)) = -\frac{1}{a}\boldsymbol{\theta}. \tag{28}$$

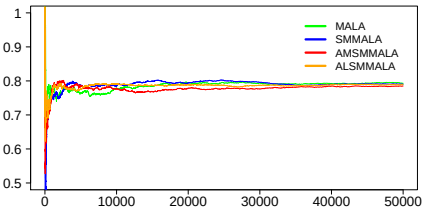
So, the gradient $\nabla_{\boldsymbol{\theta}}(\log\left(p(\boldsymbol{\theta}|\mathbf{y}, X)\right))$ of the log-target equals the sum of (27) and (28).

The metric of a log-target p is conventionally taken to be the expected Fisher information of p, which is equal to the negative Hessian of log-likelihood L minus the Hessian of log-prior π in a Bayesian setting. By differentiating (27) and (28), it is deduced that the metric $G(\boldsymbol{\theta}|\mathbf{y}, X)$ for the Bayesian logistic regression model with a normal prior $\mathcal{N}(\mathbf{0}, vI)$ is

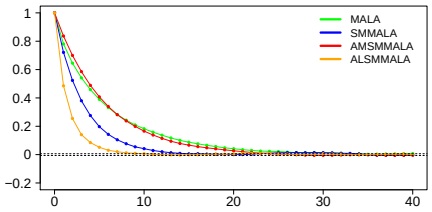
$$G(\boldsymbol{\theta}|\mathbf{y}, X)) = X^T\Lambda(\boldsymbol{\theta})X + \frac{1}{v}I, \tag{29}$$

where the $n_d \cdot n_d$ diagonal matrix $\Lambda(\boldsymbol{\theta}) = diag[p_i(1-p_i)]$ has diagonal elements

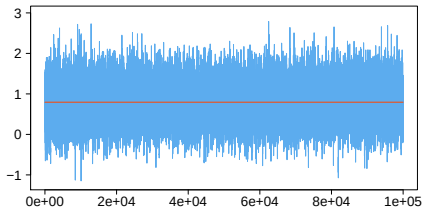
$$p_i = P(y_i = 1) = \exp(\boldsymbol{\theta}^T\mathbf{x_{i,}})/(1+\exp(\boldsymbol{\theta}^T\mathbf{x_{i,}})),\ i \in \{1, 2, \ldots, n_d\}. \tag{30}$$



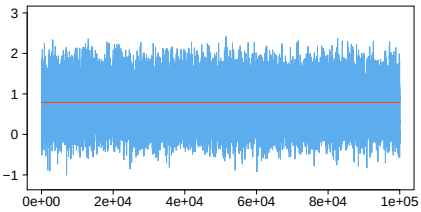
(a) Running mean



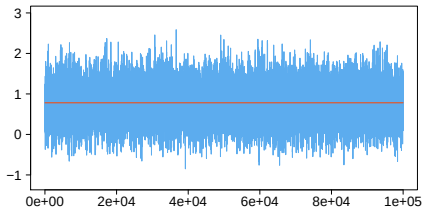
(b) Linear ACF



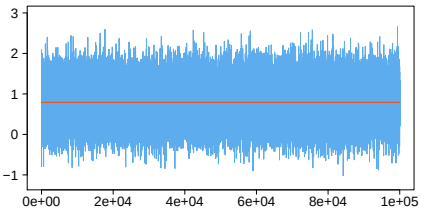
(c) MALA traceplot



(d) SMMALA traceplot



(e) AMSMMALA traceplot



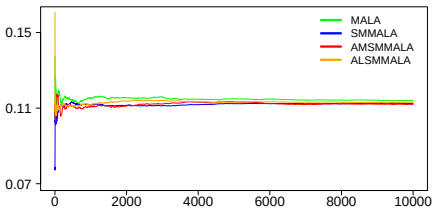
(f) ALSMMALA traceplot

Figure 4: Logistic regression - plots

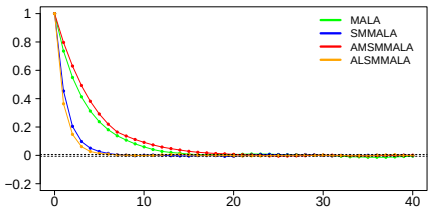
Table 3: Poisson regression - comparison of Langevin sampling methods

Method	AR	ESS				Time	Efficiency	Speedup
		θ_1	θ_2	θ_3	θ_4			
MALA	.60	12649	14519	12942	32161	2.36	5354.88	1.00
SMMALA	.71	40132	39657	39522	39552	5.48	7212.76	1.35
AMSMMALA	.26	11825	11667	11530	11820	3.12	3692.49	0.69
ALSMMALA	.60	43157	43089	42892	43249	3.83	11194.52	2.09

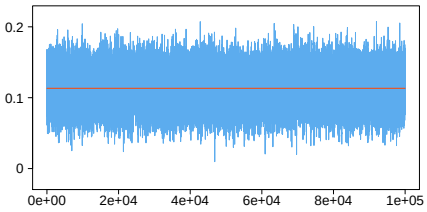
4.2 Bayesian Poisson Regression



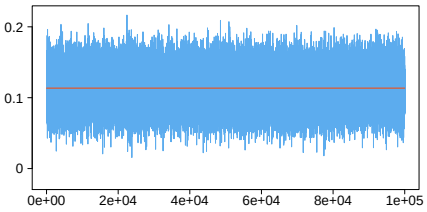
(a) Running mean



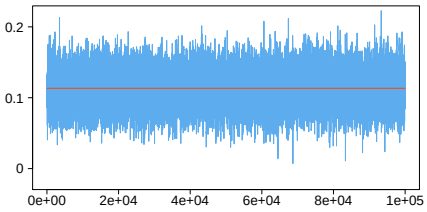
(b) Linear ACF



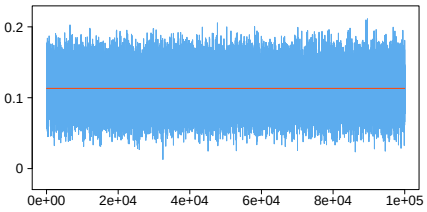
(c) MALA traceplot



(d) SMMALA traceplot



(e) AMSMMALA traceplot



(f) ALSMMALA traceplot

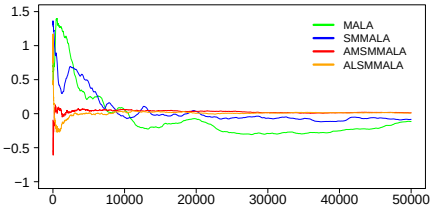
Figure 5: Poisson regression - plots

Table 4: t-density target - comparison of Langevin sampling methods. SMMALA (r) is SMMALA with reverse mode AD, and SMMALA (f) is SMMALA with forward mode AD

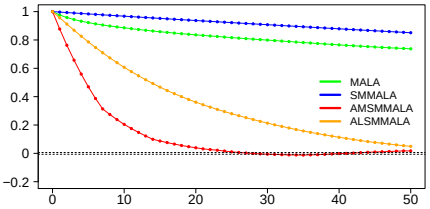
Method	AR	ESS			Time	Efficiency	Speedup
		mean	median	max			
MALA	.60	127	145	239	1.95	65.09	1.00
SMMALA (r)	.67	103	114	123	41.39	2.48	0.04
SMMALA (f)	.68	97	104	121	575.29	0.17	0.00
AMSMMALA	.16	7753	7866	7960	15.27	507.74	7.80
ALSMMALA	.96	2465	2538	2574	16.16	152.51	2.34

4.3 Multivariate t-Distribution

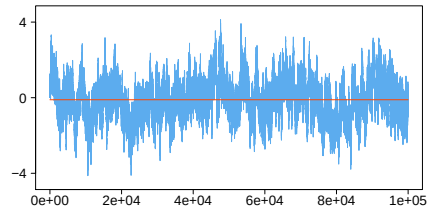
Mention that the base algorithm drives the simulation and sets the properties of the sampler. For this reason AMSMMALA works better than ALSMMALA for the t-dist example. Reference ForwardDiff conference paper, Revels et al. (2016).



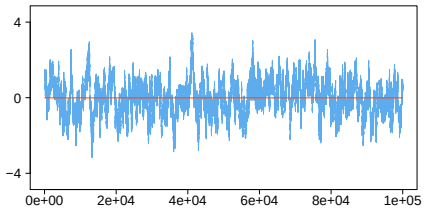
(a) Running mean



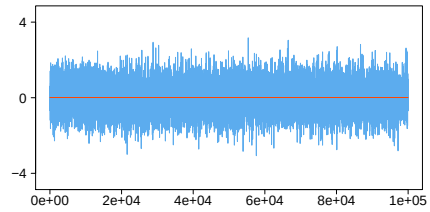
(b) Linear ACF



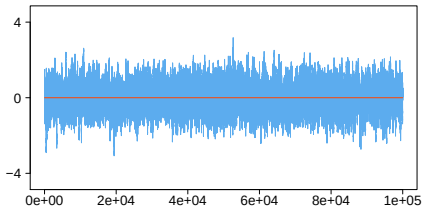
(c) MALA traceplot



(d) SMMALA traceplot



(e) AMSMMALA traceplot



(f) ALSMMALA traceplot

Figure 6: t-density target - plots

5. Discussion

Acknowledgments

We would like to acknowledge support for this project from the National Science Foundation (NSF grant IIS-9988642).

References

Michael Betancourt. *A General Metric for Riemannian Manifold Hamiltonian Monte Carlo*, pages 327–334. Springer, Berlin Heidelberg, 2013. ISBN 978-3-642-40020-9.

Simon Byrne and Mark Girolami. Geodesic monte carlo on embedded manifolds. *Scandinavian Journal of Statistics*, 40(4):825–845, 2013. ISSN 1467-9469.

Ben Calderhead and Mark Girolami. Statistical analysis of nonlinear dynamical systems using differential geometric sampling methods. *Interface Focus*, 1(821–835), 2011. ISSN 2042-8898.

Ben Calderhead, Michael Epstein, Lucia Sivilotti, and Mark Girolami. *Bayesian Approaches for Mechanistic Ion Channel Modeling*, chapter 13, pages 247–272. Humana Press, Totowa, NJ, 2013.

Siddhartha Chib and Edward Greenberg. Understanding the metropolis-hastings algorithm. *The American Statistician*, 49(4):327–335, November 1995.

Umut Şimşekli, Roland Badeau, A. Taylan Cemgil, and Gaël Richard. Stochastic quasi-newton langevin monte carlo. In *International Conference on Machine Learning (ICML)*, New York, NY, United States, June 2016.

Sandy Davie and Andrew J. Stothers. Improved bound for complexity of matrix multiplication. *Proceedings of the Royal Society of Edinburgh, Section: A Mathematics*, 143: 351–369, 4 2013. ISSN 1473-7124.

Simon Duane, A.D. Kennedy, Brian J. Pendleton, and Duncan Roweth. Hybrid monte carlo. *Physics Letters B*, 195(2):216–222, 1987. ISSN 0370-2693.

François Le Gall. Powers of tensors and fast matrix multiplication. In *Proceedings of the 39th International Symposium on Symbolic and Algebraic Computation*, ISSAC '14, pages 296–303, New York, NY, USA, 2014. ACM.

Charles J. Geyer. Practical markov chain monte carlo. *Statistical Science*, 7(4):473–483, 11 1992.

Philip E Gill, Gene H Golub, Walter Murray, and Michael A Saunders. Methods for modifying matrix factorizations. *Mathematics of Computation*, 28(126):505–535, 1974.

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. ISSN 1467-9868.

Andreas Griewank. On automatic differentiation and alalgorithm linearization. *Pesquisa Operacional*, 34:621–645, 12 2014. ISSN 0101-7438.

Andreas Griewank and Andrea Walther. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. Number 105 in Other Titles in Applied Mathematics. SIAM, Philadelphia, PA, 2nd edition, 2008. ISBN 978–0–898716–59–7.

Heikki Haario, Eero Saksman, and Johanna Tamminen. An adaptive metropolis algorithm. *Bernoulli*, 7(2):223–242, 04 2001.

Bruce Hajek. *Cooling Schedules for Optimal Annealing*, pages 147–150. Springer New York, New York, NY, 1987.

Nicholas J. Higham. Computing a nearest symmetric positive semidefinite matrix. *Linear Algebra and its Applications*, 103:103–118, 1988. ISSN 0024-3795.

Nicholas J. Higham. Computing the nearest correlation matrix - a problem from finance. *IMA Journal of Numerical Analysis*, 22(3):329–343, 2002.

Shiwei Lan, Jeffrey Streets, and Babak Shahbaba. Wormhole hamiltonian monte carlo. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, pages 1953–1959, 2014.

Shiwei Lan, Vasileios Stathopoulos, Babak Shahbaba, and Mark Girolami. Markov chain monte carlo from lagrangian dynamics. *Journal of Computational and Graphical Statistics*, 24(2):357–378, 2015.

Shiwei Lan, Tan Bui-Thanh, Mike Christie, and Mark Girolami. Emulation of higher-order tensors in manifold monte carlo methods for bayesian inverse problems. *Journal of Computational Physics*, 308:81–101, 2016. ISSN 0021-9991.

Samuel Livingstone and Mark Girolami. Information-geometric markov chain monte carlo methods using diffusions. *Entropy*, 16(6):3074, 2014. ISSN 1099-4300.

Radford M. Neal. *Bayesian Learning for Neural Networks*, volume 118. Springer, 1996.

Yaghout Nourani and Bjarne Andresen. A comparison of simulated annealing cooling strategies. *Journal of Physics A: Mathematical and General*, 31(41):8373–8385, 1998.

Marcelo Pereyra. Proximal markov chain monte carlo algorithms. *Statistics and Computing*, 26(4):745–760, 2016. ISSN 1573-1375.

Jarrett Revels, Miles Lubin, and Theodore Papamarkou. Forward-mode automatic differentiation in julia. *arXiv*, 2016.

Gareth O. Roberts and Jeffrey S. Rosenthal. Optimal scaling of discrete approximations to langevin diffusions. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 60(1):255–268, 1998. ISSN 1467-9868.

Gareth O. Roberts and Osnat Stramer. Langevin diffusions and metropolis-hastings algorithms. *Methodology And Computing In Applied Probability*, 4(4):337–357, 2002. ISSN 1573-7713.

Gareth O. Roberts and Richard L. Tweedie. Exponential convergence of langevin distributions and their discrete approximations. *Bernoulli*, 2(4):341–363, 12 1996.

Raphaela Schwentner, Theodore Papamarkou, Maximilian O. Kauer, Vassilios Stathopoulos, Fan Yang, Sven Bilke, Paul S. Meltzer, Mark Girolami, and Heinrich Kovar. Ews-fli1 employs an e2f switch to drive target gene expression. *Nucleic Acids Research*, 2015.

Matthias Seeger. Low rank updates for the cholesky decomposition. Technical report, 2004.

James H. Wilkinson. Modern error analysis. *SIAM Review*, 13(4):548–568, 1971.

Virginia Vassilevska Williams. Breaking the coppersmith-winograd barrier, 2011.