Maximum conditional entropy Hamiltonian Monte Carlo sampler

Yu, Tengchao; Wang, Hongqiao; Li, Jinglai

DOI:
10.1137/20M1341192

Document Version
Peer reviewed version

Citation for published version (Harvard):
Yu, T, Wang, H & Li, J 2021, 'Maximum conditional entropy Hamiltonian Monte Carlo sampler', *SIAM Journal on Scientific Computing*, vol. 43, no. 5, pp. A3607–A3626. https://doi.org/10.1137/20M1341192

MAXIMUM CONDITIONAL ENTROPY HAMILTONIAN MONTE CARLO SAMPLER*

TENGCHAO YU†, HONGQIAO WANG‡, AND JINGLAI LI§

Abstract. The performance of Hamiltonian Monte Carlo (HMC) sampler depends critically on some algorithm parameters such as the total integration time and the numerical integration stepsize. The parameter tuning is particularly challenging when the mass matrix of the HMC sampler is adapted. We propose in this work a Kolmogorov-Sinai entropy (KSE) based design criterion to optimize these algorithm parameters, which can avoid some potential issues in the often used jumping-distance based measures. For near-Gaussian distributions, we are able to derive the optimal algorithm parameters with respect to the KSE criterion analytically. As a byproduct the KSE criterion also provides a theoretical justification for the need to adapt the mass matrix in HMC sampler. Based on the results, we propose an adaptive HMC algorithm, and we then demonstrate the performance of the proposed algorithm with numerical examples.

Key words. Hamiltonian Monte Carlo, Kolmogorov-Sinai entropy, Markov chain Monte Carlo.

1. Introduction.

Generating samples from a target distribution is an important practice in many fields of science and engineering. The Hamiltonian Monte Carlo (HMC) method [20, 2], initially developed in the Physics community [10], has become a very popular tool for sampling distributions with continuously differentiable density functions. The HMC method receives considerable attention from Scientific Computing and Computational Statistics, and many significant improvements of the method have been been developed in the last decade, for example, [5, 7, 12, 15, 24, 26, 28], just to name a few. Loosely speaking, the HMC method proposes new samples by simulating particle movement of a Hamiltonian system constructed from the target density for a given amount of time, and it can often achieve better performance than standard random-walk based Markov chain Monte Carlo (MCMC) algorithms as it takes advantage of the gradient information of the target distribution [20, 2].

It is well-known that the performance of the HMC algorithm depends critically on the algorithm parameters, and in particular on the mass matrix or the metric [4], the total integration time and the numerical integration stepsize. As such, tuning these parameters becomes an important task in the implementation of HMC. First, a popular practice to tune the mass matrix is to set it to be the inverse of the covariance of the target distribution, and the physical intuition behind this choice can be found in e.g., [20]. A rigorous justification of such a choice has yet been established to the best of our knowledge. On the other hand, a number of methods have been developed to tune the other two key algorithm parameters: the total integration time and the integration stepsize. Such works include, the optimal tuning derived in the infinite dimensional limit [3], the No U-turn sampler (NUTS) [15], and the Adaptive HMC sampler (AHMC) [26]. However, it is rather difficult to directly apply these methods when the mass matrix is adapted during the MCMC iteration: for example, in the implementation of NUTS in STAN [6] the mass matrix is estimated in the warmup period and fixed afterwards. The main problem concerned in this work is how to determine the

*The work was supported by the NSFC, under grant number 11301337.

†School of Mathematical Sciences, Shanghai Jiao Tong University, Shanghai 200240, China, (tengchaoyu@sjtu.edu.cn).

‡School of Mathematics and Statistics, Central South University, Changsha, Hunan 410083, China, (wanghongqiao@csu.edu.cn).

§Corresponding Author, School of Mathematics, University of Birmingham, Birmingham B15 2TT, UK, (j.li.10@bham.ac.uk).

integration time and the numerical stepsize when the mass matrix is adapted.

When designing a tuning scheme of MCMC, one usually relies on a specific design criterion or performance measure, and for example, in both [3, 26], the tuning schemes make use of the expected squared jumping distance (ESJD), which was proposed in [22] for random walk MCMC algorithms. The ESJD criterion, which seeks the largest jumping distance, may cause some potential issues in the HMC sampler. An intuitive example for this is that the particles jumps from one side of the space to the other in each iteration, and in this case even though they may travel a very long distance, the Markov chain may not even be ergodic [20]. In Section 2.3 we shall demonstrate that this is exactly the case when the HMC proposal is tuned by maximizing the ESJD for the simple Gaussian target distributions. To address the issue we propose a new design criterion based on the Kolmogorov-Sinai entropy (KSE) [23] of the underlying Markov chain and we show that this criterion can lead to the very optimal proposal in the Gaussian case (namely, when the target distribution is exactly Gaussian, the proposal becomes the target distribution itself). Moreover, for near-Gaussian distributions, we can analytically derive the optimal integration time with respect to the KSE (which, as will be shown in Section 2, is essentially the conditional entropy of the underlying Markov chain) criterion, and this result enables us to develop a HMC algorithm that adapts the mass matrix. Interestingly we are also able to show that the optimal mass matrix under the KSE criterion is actually the inverse of the target covariance, providing a justification of the strategy previously obtained from physical intuition [20]. With numerical examples, we show that the proposed algorithm has a rather competitive performance against existing methods for a certain class of problems, i.e., those with near-Gaussian distributions.

To summarize, the *main contribution* of the present work is two fold: first we provide a KSE based design criterion to determine the algorithm parameters of HMC proposal, and for near-Gaussian distributions, we derive an analytical result of the optimal integration time under the KSE criterion; second, using the optimality results we design an adaptive HMC algorithm which automatically determines the stepsize while the mass matrix is adapted. The rest of the paper is organized as follows. In Section 2 we present the proposed Maximum conditional entropy (or KSE) HMC sampler, and provide some optimality results of the sampler. In Section 3 we demonstrate the performance of the proposed algorithm with mathematical and practical examples. Finally in Section 4, we provide some concluding remarks on both the advantages and the limitations of the proposed method.

2. The maximum conditional entropy HMC sampler.

2.1. The HMC algorithm.

Let $\mathbf{x}$ be a n-dimensional random variable with distribution

$$\pi(\mathbf{x}) = \exp(-U(\mathbf{x})). \tag{2.1}$$

By using the Hamiltonian dynamics, one can design a very efficient scheme to draw samples from the distribution $\pi(\mathbf{x})$. Specifically one constructs an artificial Hamiltonian system that has $\mathbf{x}$ as its position variable and the function $U(\mathbf{x})$ as the potential energy of the system, and then introduces an auxiliary variable $\mathbf{p}$ to be the momentum of the system with kinetic energy $K(\mathbf{p})$. In practice, the kinetic energy is usually taken to be of a quadratic form:

$$K(\mathbf{p}) = \frac{1}{2}\mathbf{p}^T M^{-1}\mathbf{p}, \tag{2.2}$$

Algorithm 1 The leapfrog algorithm

function $(\mathbf{x}_T, \mathbf{p}_T)$=LEAPFROG$(\mathbf{x}_0, \mathbf{p}_0, M, U, \epsilon, L)$
  for $i = 1$ to L **do**
    Set $\mathbf{p} \leftarrow \mathbf{p}_0 + \frac{1}{2}\epsilon\nabla U(\mathbf{x}_0)$;
    Set $\mathbf{x} \leftarrow \mathbf{x}_0 + \epsilon M^{-1}\mathbf{p}$
    Set $\mathbf{p} \leftarrow \mathbf{p} + \frac{1}{2}\epsilon\nabla U(\mathbf{x})$;
  end for
  Set $\mathbf{x}_T \leftarrow \mathbf{x}$ and $\mathbf{p}_T \leftarrow \mathbf{p}$;
end function

where M is a positive definite symmetric matrix. The dynamics of the constructed system is governed by

$$\frac{d\mathbf{x}}{dt} = \frac{\partial H}{\partial \mathbf{p}}, \quad \frac{d\mathbf{p}}{dt} = -\frac{\partial H}{\partial \mathbf{x}}. \tag{2.3}$$

Suppose the current position is $\mathbf{x}_0$ and the HMC performs the following steps in each iteration,

- Sample an initial momentum state $\mathbf{p}_0$ from the distribution $N(0, M)$;
- Solve the Eq. (2.3) with initial condition $(\mathbf{x}_0, \mathbf{p}_0)$, for a given amount of time T, obtaining the new states $(\mathbf{x}_T, \mathbf{p}_T)$;
- Accept the new position $\mathbf{x}_T$ with probability

$$\min[1, \exp(H(\mathbf{x}_0, \mathbf{p}_0) - H(\mathbf{x}_T, \mathbf{p}_T))]. \tag{2.4}$$

Since the Hamiltonian system preserves its total energy,

$$H(\mathbf{x}(t), \mathbf{p}(t)) = U(\mathbf{x}(t)) + K(\mathbf{p}(t)),$$

it should be clear that if we can solve the Eq. (2.3) exactly, the acceptance probability is simply one. In practice, however, Eq. (2.3) must be solved numerically and as a result the acceptance probability is lower than one. The leapfrog algorithm [20] is commonly used to solve the system for its ability to preserve the time-reversibility, and moreover, when the integration time T is large, one often use multiple leapfrog steps to integrate the ODE system (2.3) from 0 to T. In Alg. 1 we present the multiple-step leapfrog algorithm for solving the ODE system (2.3).

2.2. Maximizing the entropy rate. Now suppose that we can solve the Hamiltonian system (2.3) exactly, achieving the 100% acceptance probability, and an important question here is *how to choose matrix M and time T for the best efficiency of the HMC algorithm?* To answer this question, we first need to establish an optimality criterion or a performance measure of the HMC sampler.

A very natural choice for such a performance measure is the ESJD:

$$\mathbb{E}[\|\mathbf{x}_T - \mathbf{x}_0\|^2], \tag{2.5}$$

which was first proposed by Pasarica and Gelman [22] and was later applied to HMC in [3, 26]. As is mentioned earlier the main idea behind the ESJD criterion is to seek a proposal that moves the largest distance from the present location, and this idea performs well in a number of target distributions. However, as will be shown later, maximizing the ESJD may lead to problematic HMC

proposals in certain circumstances and in particular, in the simple Gaussian case, the resulting chain becomes periodic and loses its ergodicity. Naturally we expect that the ESJD criterion may not perform well for distributions that are close to Gaussian, an important class of distributions in many practical problems. This issue then motivates us to consider alternative design criterion. It is well known that a general principle to design a MCMC scheme is to minimize its mixing time, which however can not be directly used as it is extremely difficult to compute. However, it is reported in [18] that maximizing the KSE can yield very close results to those of minimizing the mixing time for general Markov chains. Based on this finding we propose to determine the parameters in HMC by maximizing the KSE. The intuition for maximizing the KSE can also be understood as to seek a transition distribution that is maximally dispersing (as is measured by KSE) so that it can explore the state space efficiently, an idea often used in the network theory [8]. For a stationary time-reversible Markov chain, the KSE is equal to the conditional entropy (CE):

$$\mathbb{H}(\mathbf{x}_T|\mathbf{x}_0) = \int \log \pi(\mathbf{x}_T|\mathbf{x}_0)\pi(\mathbf{x}_T, \mathbf{x}_0)d\mathbf{x}_T d\mathbf{x}_0, \tag{2.6}$$

which means that the algorithm parameters are determined by maximizing the CE of the chain. When $\pi(\mathbf{x}_T|\mathbf{x}_0)$ is Gaussian, the CE in Eq. (2.6) is (up to a constant) equal to

$$\mathbb{E}_{\mathbf{x}_0}[\log\det \mathrm{Cov}[\mathbf{x}_T|\mathbf{x}_0]]. \tag{2.7}$$

Thus when $\pi(\mathbf{x}_T|\mathbf{x}_0)$ is Gaussian or near-Gaussian, we can choose to optimize Eq. (2.7) for that it is usually easier to evaluate than Eq. (2.6).

2.3. Near-Gaussian distributions. In this section we consider the situation where the target distribution is near-Gaussian. The rationale here is that we can first derive the optimal algorithm parameters including T by assuming the target distribution is exactly Gaussian, and it should be sensible to use derived parameter values for the actual target distribution provided it does not deviate too far from Gaussian. Such near-Gaussian distributions arise frequently in Bayesian inference problems, especially for those with a large amount of data, thanks to the asymptotic normality property [11]. Following this idea we shall assume that the target distribution can be well approximated by a multivariate Gaussian $\pi(\mathbf{x}) \approx \mathcal{N}(\mu, \Sigma)$, where μ and Σ are the mean and the covariance of $\pi(\mathbf{x})$ respectively. Without loss of generality we shall assume $\mu = 0$ to keep the calculations simple.

The main result. In this case, we aim to determine the following algorithm parameters: the momentum covariance matrix M, and the integration time T. Here for the convenience of theoretical analysis, we need to impose a constraint on M: it commutes with Σ. That is, we shall choose M from $\mathcal{M}_\Sigma^+$, the class of all positive definite matrices that commute with Σ. As is discussed in Section 2.2 these parameters will be determined by solving

$$\max_{\{M\in\mathcal{M}_\Sigma^+, T\in R^+\}} \mathbb{E}_{\mathbf{x}_0}[\log\det \mathrm{Cov}[\mathbf{x}_T|\mathbf{x}_0]], \tag{2.8}$$

where R^+ denotes all positive real numbers. The following theorem states the main result of the work:

THEOREM 2.1. *Suppose that* $\pi(\mathbf{x}) = \mathcal{N}(0, \Sigma)$, $\mathbf{x}_0 \sim \pi(\mathbf{x})$, $\mathbf{p}_0 \sim \mathcal{N}(0, M)$, *and* $\mathbf{x}(t)$ *is the solution of the Hamiltonian system* (2.3) *with* $K(\mathbf{p})$ *given by Eq.* (2.2). *Let* $\mathbf{x}_T = \mathbf{x}(T)$, *and a solution of the optimization problem* (2.8) *is*

$$M = \Sigma^{-1}, \quad \text{and} \quad T = (2m+1)\pi/2, \tag{2.9}$$

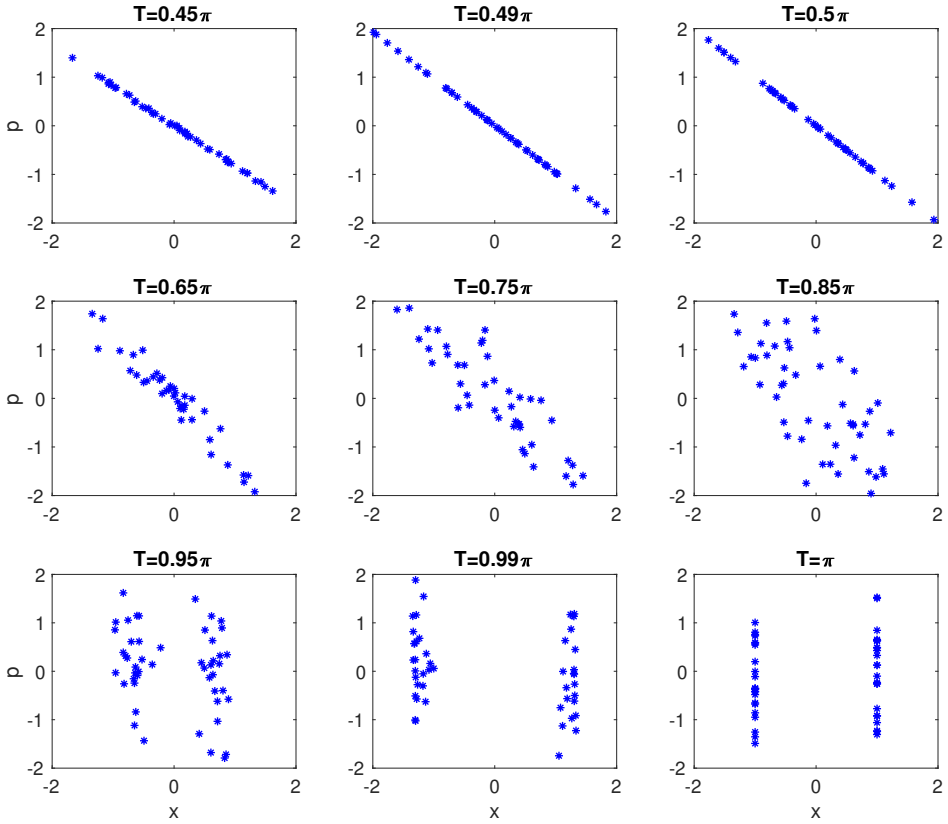

FIG. 2.1. *The scatter plots of the samples in the (x, p) space. Top: the results of the HMC with three different values of T near the optimal value with respect to CE; Middle: the results of the HMC with three different values of T between 0.5π and π; Bottom: the results with three different values of T near the optimal value with respect to ESJD.*

for an arbitrary non-negative integer m.

Proof: First we define $A = M^{-1}\Sigma^{-1}$, and since M^{-1} and Σ^{-1} commutes, A is also a positive definite matrix. Now suppose that we conduct an eigenvalue decomposition of A, yielding $A = V\Lambda V^{-1}$, where V is a square matrix whose the i-th column is the eigenvector $\boldsymbol{v}_i$ of $\boldsymbol{A}$ and $\boldsymbol{\Lambda}$ is the diagonal matrix whose diagonal elements are the corresponding eigenvalues, $\boldsymbol{\Lambda}_{i,i} = \lambda_i$.

Under the assumption that the target distribution $\pi(\mathbf{x}) = \mathcal{N}(0, \Sigma)$, the Hamiltonian system (3)

can be solved analytically,

$$\boldsymbol{x}(t) = \sum_{i=1}^{n}(a_i \cos\sqrt{\lambda_i}t + b_i \sin\sqrt{\lambda_i}t)\boldsymbol{v}_i, \tag{2.10a}$$

$$\boldsymbol{p}(t) = M\sum_{i=1}^{n}\sqrt{\lambda_i}(-a_i \sin\sqrt{\lambda_i}t + b_i \cos\sqrt{\lambda_i}t)\boldsymbol{v}_i, \tag{2.10b}$$

where the coefficients $\{a_i, b_i\}_{i=1}^n$ are determined via the initial conditions:

$$\boldsymbol{x}_0 = \sum_{i=1}^{n} a_i\boldsymbol{v}_i, \quad \boldsymbol{p}_0 = M\sum_{i=1}^{n}\sqrt{\lambda_i}b_i\boldsymbol{v}_i. \tag{2.10c}$$

By some elementary calculations, we obtain from Eqs. (2.10) that,

$$C = \mathrm{Cov}(\boldsymbol{x}_T|\boldsymbol{x}_0) = \sum_{i=1}^{n}\sum_{j=1}^{n}\mathbb{E}[b_i b_j]\sin(\sqrt{\lambda_i}T)\sin(\sqrt{\lambda_j}T)\boldsymbol{v}_i\boldsymbol{v}_j^T. \tag{2.11}$$

Now using Eq. (2.10c) and the facts that $\boldsymbol{x}_0 \sim \mathcal{N}(0,\Sigma)$ and $\mathbf{p}_0 \sim \mathcal{N}(0,M)$, we derive that $(b_1,...,b_n)$ follows a multivariate Gaussian distribution and

$$\mathbb{E}[b_i, b_j] = \Gamma_{i,j}, \quad \text{where} \quad \Gamma = \Lambda^{-\frac{1}{2}}V^{-1}M^{-1}V^{-T}\Lambda^{-\frac{1}{2}}, \tag{2.12}$$

for any $0 \le i, j \le n$.

Substituting Eq. (2.12) into Eq. (2.11) yields,

$$C = V\Lambda_{\sin}\Gamma\Lambda_{\sin}V^T, \tag{2.13}$$

with

$$\Lambda_{\sin} = \mathrm{diag}[\sin\sqrt{\lambda_1}T, ..., \sin\sqrt{\lambda_n}T]. \tag{2.14}$$

It follows that

$$\det C = \det(V\Lambda_{\sin}\Gamma\Lambda_{\sin}V^T) = \prod_{i=1}^{n}\lambda_{\Sigma,i}\frac{1-\cos 2\sqrt{\lambda_i}T}{2}, \tag{2.15}$$

where $\lambda_{\Sigma,1}, ..., \lambda_{\Sigma,n}$ are the eigenvalues of Σ. Now recall that we want to find a solution of

$$\begin{aligned}&\max_{\{M\in\mathcal{M}_\Sigma^+, T\in R^+\}} \mathbb{E}_{\mathbf{x}_0}[\log\det\mathrm{Cov}[\mathbf{x}_T|\mathbf{x}_0]]\\ =&\max_{\{M\in\mathcal{M}_\Sigma^+, T\in R^+\}}\sum_{i=1}^{n}\log\frac{1-\cos 2\sqrt{\lambda_i}T}{2} + \log\det\Sigma.\end{aligned} \tag{2.16}$$

It is obvious that the maximum of the problem is attained at

$$\sqrt{\lambda_i}T = \frac{(2m+1)}{2}\pi, \quad \text{for} \quad i = 1...n,$$

where m is an arbitrary integer. A special choice is that

$$T = \frac{(2m+1)}{2}\pi,$$

and $\lambda_i = 1$ for $i = 1...n$. It follows immediately that matrix A is identity, which implies that $M = \Sigma^{-1}$. □

In practice, since the Hamiltonian system needs to be solved numerically, it is certainly desirable to use smaller integration time T, and thus in the HMC algorithm we set $m = 1$ and $T = \pi/2$. We reinstate here that, while the trick to improve the efficiency of HMC by choosing $M = \Sigma^{-1}$ has long been known from an intuitive perspective [20], we are able to provide a justification for it based on the maximum CE (MCE) principle.

Comparison with ESJD: a univariate example. Here we use a simple univariate example to demonstrate the difference between the CE criterion and the ESJD. Namely we assume that the target distribution is $N(0, k)$, $K(p) = p^2/(2m)$ and $p_0 \sim N(0, m)$. In this case, by some elementary calculations we can derive that an integration time that maximizes ESJD (2.5) is $T = \sqrt{km}\pi$, and the associated proposal is: $x_T = -x_0$, which means that the samples will only jump between two locations (x_0 and $-x_0$) and the resulting chain is certainly not ergodic. On the other hand, the optimal integration time with respect to CE is $T = \sqrt{km}\pi/2$, and the associated proposal is $\pi(x_T|x_0) = N(0, k)$, i.e., to sample directly from the target distribution, which is the very optimal distribution in this case. We refer to Appendix A for a detailed derivation of the optimal integration time with respect to KSE and ESJD. It is also easy to derive that, the efficiency of the proposal behaves periodically with respect to T in this case. In the first numerical example, we use numerical experiments to validate the theoretical analysis conducted here.

Determining the number of leapfrog steps. Now we have derived the optimal integration time T, and as is discussed earlier, we need to integrate the Hamiltonian system (2.3) from 0 to T using the leapfrog algorithm. In particular, we usually need to perform leapfrog integration multiple times to achieve the necessary numerical precision, and to this end, the number of leapfrog steps, conventionally denoted by L, is another key algorithm parameter to be specified. If we increase L, the numerical integration becomes more accurate and the acceptance probability approaches to 1, and the price to pay is that more leapfrog steps means higher computational cost. We note here that in most existing works both L and $\epsilon = T/L$ need to be determined simultaneously, and in that case higher acceptance probability does not necessarily imply a better proposal even without considering the computational cost for computing the proposal. In our method, however, since the total integration time T is fixed, it is reasonable to assume that increasing L, which in turn increases the acceptance probability, improves the performance of the algorithm. Based on this idea, we shall seek the value of L that provides the highest acceptance rate per computational cost (which is usually measured by L). Namely, we use an adaptive scheme to gradually increase L until the average acceptance rate per L does not improve.

Adaptively estimating the covariance matrix. Another important issue in the method is that it requests the knowledge of the target covariance matrix. Here we follow the idea of the adaptive MCMC algorithms to estimate the covariance from the sample history and specifically the target covariance is updated using the method given in [14, 13]. We emphasize here that, the adaptive algorithm does not require an accurate estimation of the target covariance in advance (i.e. from a pilot runs); rather it adaptively improves the estimate of the covariance as the sample size increases [1]. We present the complete algorithm in Alg. 2, where the main iteration (from step 6 to step 49) consists of three major tasks: conducting a HMC iteration, estimating covariance matrix

Algorithm 2 Maximum Conditional Entropy Sampler

Require: $U(\mathbf{x})$, Acc_{min}, N_0, L_0, L_{max}, N_{max}, N_M, N_L, ρ, I_{max}

Initialization: draw N_0 samples $\{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_{N_0}\}$ using standard HMC sampler.
Estimate the sample covariance matrix $\hat{\Sigma}$ of $\{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_{N_0}\}$;
Let $M = \hat{\Sigma}^{-1}$;
Let $T = \pi/2$;
Let $Acc_{old} = 0$, $L_{old} = L_0$, $L = L_0$, and $I_L = 1$,$I_M = 0$;
for $t = N_0$ to N_{max} **do**
    $\epsilon = T/L$;
    Draw $\mathbf{p}_t \sim N(0,\ M)$;
    $(\mathbf{x}^*,\ \mathbf{p}^*) = \text{leapfrog}(\mathbf{x}_t, \mathbf{p}_t, M, U(\mathbf{x}), \epsilon, L)$;
    Draw $u \sim U(0, 1)$;
    if $u < \min\{1, \exp(H(\mathbf{x}_t, \mathbf{p}_t) - H(\mathbf{x}^*, \mathbf{p}^*))\}$ **then**
        $\mathbf{x}_{t+1} = \mathbf{x}^*$;
    else
        $\mathbf{x}_{t+1} = \mathbf{x}_t$;
    end if
    if $t \mod N_L = 0$ **then**
        Let Acc be the average acceptance probability of the last N_L samples;
        if $t < N_{\text{M}}$ and ($I_M = 1$ or $Acc > 0$) **then**
            Update the sample covariance matrix $\hat{\Sigma}$ with the last N_L samples;
            Let $M = \hat{\Sigma}^{-1}$;
            $I_M = 1$;
        end if
        if $I_L = 1$ **then**
            if $L = L_{max}$ **then**
                $I_L = 0$;
                if $Acc/L < Acc_{old}/L_{old}$ **then**
                    $L = L_{old}$;
                end if
            else
                if $Acc > Acc_{min}$ **then**
                    if $Acc/L < Acc_{old}/L_{old}$ **then**
                        $I_{count} = I_{count} + 1$;
                        if $I_{count} \geq I_{max}$ **then**
                            $I_L = 0$, $L = L_{old}$;
                        end if
                    else
                        $Acc_{old} = Acc$, $L_{old} = L$;
                        $I_{count} = 0$;
                        $L = \min\{[\rho L_{old}], L_{max}\}$
                    end if
                else
                    $Acc_{old} = Acc$, $L_{old} = L$;
                    $I_{count} = 0$;
                    $L = \min\{[\rho L_{old}], L_{max}\}$.
                end if
            end if
        end if
    end if
end for

M, and determining the number of leapfrog steps L. We provide some specific remarks on the algorithm in the following:

- From step 7 to step 15, the algorithm conducts a usual HMC iteration:
  - In step 9, the function leapfrog$(\mathbf{x}_t, \mathbf{p}_t, M, U(x), \epsilon, L)$ represents to solve the Hamiltonian system (2.3) specified by M and $U(\mathbf{x})$ from the initial condition $\mathbf{x}_t$ and $\mathbf{p}_t$, using the leapfrog algorithm with stepsize ϵ for L steps.
- From step 16 to step 22, the algorithm adapts the covariance matrix:
  - In step 18, we fix the maximum number of iterations in which we update the parameters to be N_M.
  - Steps 19 and 20 update matrix M from samples, and the formulas for these computations are Eqs. (2) and (3) in [14] respectively;
- From step 23 to step 47, the algorithm determines the value of L:
  - In step 23, I_L is an indicator controlling whether L will be updated: namely the updating of L will be terminated if I_L is switched to 0.
  - From steps 24 to 28, the algorithm deals with the situation that L reaches a prescribed maximum value and in this case I_L is set to 0.
  - Step 30 examines if the current acceptance probability Acc reaches a minimal required value Acc_{min}:
    - if Acc does not reaches the minimal value Acc_{min}, L will be increased until it does, which is performed in steps 42 to 44.
    - if Acc is larger than Acc_{min}, L is iteratively increased if Acc/L continues to increase and the updating is terminated (by setting $I_L = 0$) if Acc_{min} is not increasing for a number (denoted by I_{max}) of successive steps; this procedure corresponds to steps 30 to 40.
  - In steps 39 and 44, we update L by multiplying the current value of it by a factor ρ.

3. Numerical examples. In this section, we provide four examples to demonstrate the performance of the proposed MCE sampler (MCES). The purpose of the first example to demonstrate that the failure of the ESJD criterion for the simple Gaussian distribution while MCE performs well; the second example is used to test how the method performs at several different levels of non-Gaussianity; finally the last two examples provide real-world testbeds in which we compare the performance of the MCE method and NUTS. The code of the proposed MCE method and the examples provided here are available at the Github repository [27].

3.1. Toy Problem: Univariate Gaussian. We first use a Gaussian example to compare the CE criterion with ESJD. Specifically we take target distribution to be the univariate standard Gaussian distribution $N(0, 1)$, the kinetic energy to be $K(p) = p^2/2$, and p_0 to be standard Gaussian as well. As has been discussed, the optimal integration time T computed by the proposed MCE method is $\pi/2$, while that computed by ESJD is π. We thus take six different values of T: $T = 0.45\pi$, 0.49π, 0.5π, which are close to the optimal value by MCE, and $T = 0.95\pi, 0.99\pi$, π, which are close to the optimal value predicted by ESJD. In all the tests we fix the starting position to be $x = 1$, and since for this toy problem, we have the analytical solution (see Appendix A) and so we do not need to use leapfrog. To demonstrate the behavior of the proposals, we plot the first 100 samples produced by each proposal in Fig. 2.1. First it can be seen here that if we use $T = \pi$ which is exactly the optimal value predicted by ESJD, the samples are fixed at $x = -1$ and $x = 1$ and in this case the sampler fails completely. The samples deviate away from the two points as T is slightly perturbed from π, but most samples are still concentrated near the two locations, indicating poor

performance of the proposals. On the other hand, the samples drawn by the proposals determined with the MCE method distribute well according to the target distribution, which demonstrates that the integration time predicted by the MCE method does lead to good proposals in this Gaussian example.

Parameter	L_0	L_{max}	ρ	Acc_{min}	N_{max}
Value	1	60	1.2	60%	10000
Parameter	N_L	N_M	N_0	I_{max}	
Value	200	2000	1000	1	

TABLE 3.1
Algorithm parameters of MCES.

3.2. Toy Problem: Rosenbrock function. Our second example is the Rosenbrock function, an often used benchmark problem for MCMC methods. Specifically, we here use a slightly modified version of the Rosenbrock function:

$$\pi(x_1, x_2) \propto \exp(-x_1^2 - 100(x_2 - bx_1^2)^2), \tag{3.1}$$

where the modification allows us to control the "non-Gaussianity" of the distribution. Specifically the distribution is exactly Gaussian for $b = 0$ and it departs away from Gaussian if we increase the value of b. In Fig. 3.1 (Top), we show the distributions for $b = 0.1, 0.35$ and 0.7 respectively, where we can see that the distribution departs more from Gaussian as b increases, and it becomes significantly non-Gaussian for b is larger than 0.35. Here we shall compare the performance of MCES and NUTS for various values of b ranging from 0.05 to 0.7, and evaluate how the non-Gaussianity affacts the performance of MCES. To do so, for each method we repeat the simulations 100 times, and in each simulation we draw 10000 samples with additional 1000 samples use in the burn-in period (the setup is also used in Examples 2 and 3). The algorithm parameters of MCES used in this examples and the following two are shown in Table 3.1. In this work NUTS is implemented using the Matlab package [21] by Nishimura. We then compute the Effective Sample Size (ESS) per L of each simulation, which is an often used performance measure of MCMC algorithms. The ESS is also computed using the code in [21]:

$$ESS = \frac{n}{(1 + 2\sum_{k=1}^{+\infty} \rho(k))},$$

where n is the number of samples and $\rho(k)$ is the auto-correlation of lag k. For better graphical illustration of the performance, we define the following performance ratio: namely suppose that the average ESS/L of MCES and NUTS are respectively E_1 and E_1' for dimension x_1, and E_2 and E_2' for x_2, and the performance ratio is calculated as,

$$\text{Ratio} = \frac{1}{2}(E_1/E_1' + E_2/E_2').$$

It should be clear that the performance ratio being larger than 1 indicates that MCES has better performance than NUTS in terms of ESS/L, and it being less than one indicates the opposite. The performance ratio is plotted as a function of b in Fig. 3.1 (bottom), and from the figure we can see that the ratio is considerably larger than one in the range around 0.05 to 0.3, and is still close to

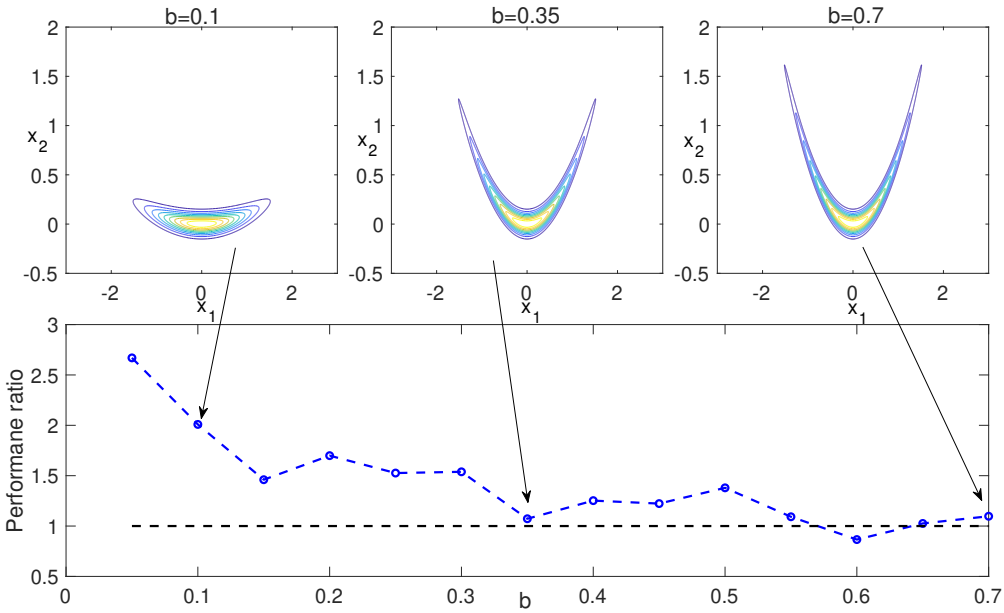

FIG. 3.1. *Top: the Rosenbrock distributions with* $b = 0.1, 0.35,$ *and* 0.7 *from left to right. Bottom: The combined ESS/L ratio plotted against the value of* b*. The black dashed line indicates a reference where the ratio is 1.*

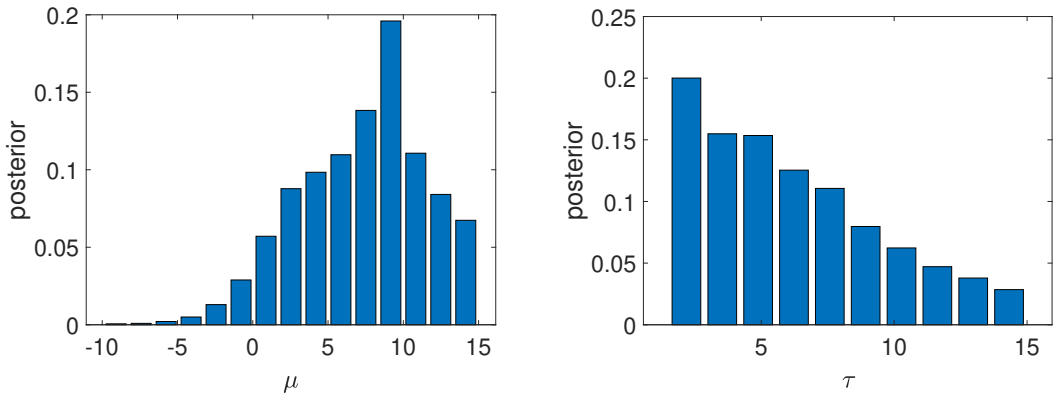

FIG. 3.2. *The histograms of the posterior samples for* μ *(left) and* τ *(right).*

one for b from 0.3 to 0.7. The results suggest that the performance of MCES does decrease as the target distribution becomes more non-Gaussian; however we also can see that MCES still yields comparable performance of NUTS in that regime, suggesting MCES may possibly be applied to problems that are considerably non-Gaussian. That said, we acknowledge that the performance of the method for strongly non-Gaussian distributions should be more throughly tested.

	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6	θ_7	θ_8	μ	τ
MCES	10.3	7.5	6.0	7.3	5.0	6.0	10.0	7.8	7.3	5.7
	(7.1)	(5.8)	(6.9)	(6.1)	(5.9)	(6.2)	(6.1)	(7.0)	(4.2)	(3.6)
NUTS	10.1	7.4	6.0	7.2	5.1	6.0	9.8	7.7	7.2	5.5
	(7.0)	(5.8)	(6.8)	(6.0)	(5.8)	(6.1)	(6.1)	(6.8)	(4.2)	(3.7)

TABLE 3.2

The posterior mean and the standard deviation (in parenthesis) computed by MCES and NUTS for the Eight School example.

3.3. Eight School problem. Our third example is the Eight School problem in [11], which is a hierarchical Bayesian inference application. While referring interested readers to [11], we here omit all the application background and proceed directly to the mathematical setup of the problem. Specifically, let $\{\theta_1, ..., \theta_8\}$ be the parameters of interest, and $\{(y_1, \sigma_1), ..., (y_8, \sigma_8)\}$ be the data. Let μ and τ be the hyperparameters specifying the prior of $\theta_1, \ldots, \theta_8$. The hierarchical model is:

$$\mu \sim \text{Uniform[-15,15]}, \quad \tau \sim \text{Uniform[0,15]}$$
$$\theta_i \sim \mathcal{N}(\mu, \tau), \quad y_i \sim \mathcal{N}(\theta_i, \sigma_i), \quad i = 1...8.$$

We use the same data as those in [11] in the inference problem, and we sample the posterior distribution with the MCE sampler and the NUTS. First to validate the proposed method, we draw 10^5 samples from the posterior distribution using both the MCE sampler and NUTS, and the posterior means and variances for all the parameters obtained by both methods are reported in Table 3.2, from which we can see that the results computed by both methods agree well with each other, up to certain statistical errors. We then plot the obtained posterior histograms for μ and τ in Fig. 3.2, and we can see from the figure that both distributions are significantly apart from Gaussian. Next we compare the performance of the MCE sampler and NUTS. We compute the ESS/L for the results of each simulation, which is used as a performance measure of the samplers, and we show the box-plots of the ESS/L results in Fig. 3.3. The plots show that the MCE sampler achieves evidently higher ESS per L than NUTS, even for the two dimensions that are evidently non-Gaussian. Additional test results (using different algorithm parameters) are provided in Appendix B, and the results demonstrate similar performance to those presented in this section, suggesting that MCES is rather robust against different values of algorithm parameters.

3.4. Bayesian Logistic regression with the German credit data. Our last example is the German credit data available at [9], a popular benchmark problem for Logistic regression. Simply put, this problem aims to classify people described by a set of attributes as good or bad credit risks. Here we use the modified version with all numerical attributes [9], which has 1000 instances each with a 24-dimensional numerical input and a binary output. For further details of the dataset, please refer to the description of the data set at [9]. Here the problem is formulated as a Logistic regression [16] and the regression coefficients $\boldsymbol{\beta} = (\beta_0, \beta_1, ..., \beta_n)$ with $n = 24$ are estimated with a Bayesian inference. The prior distribution is chosen to be standard Gaussian: $\boldsymbol{\beta} \sim \mathcal{N}(0, I)$. Just like the previous example, for each method we repeat the simulations 100 times, and in each simulation we draw 10000 samples with additional 1000 samples used for burn-in. The algorithm parameters are the same as those used in Example 2. To validate the MCES method, we have verified that the posterior means and variances computed by both methods agree well with each other. Specifically in Table 3.3 we present the posterior means and variances of all the parameters

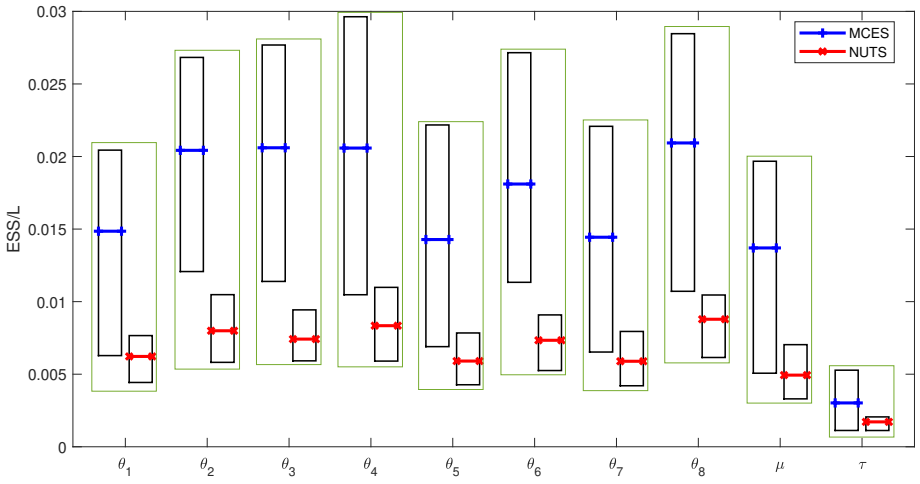

FIG. 3.3. *The box plots of the ESS per L for both the MCE sampler (MCES) and the NUTS.*

computed by MCES and NUTS. Just like the Eight School example, the results of the two methods agree well with each other, up to certain statistical errors.

We estimate the ESS/L of each dimension for each simulation, and we then compute the mean and the standard deviation of the 100 trials. We show the results in Table 3.4, which show that the average ESS/L of MCES is higher than twice of that of NUTS in all the 25 dimensions. These results demonstrate that the MCES method has a good performance in the Bayesian Logistic regression, a class of often-encountered real world problems. The good performance of the MCES method in such problems is somehow as expected, since the posteriors in the Bayesian logistic regression usually do not deviate vastly from Gaussian [17].

3.5. Log-Gaussian Cox Process. The Log-Gaussian Cox process (LGCP) is a widely used model for spatial point process data [19]. Mathematically it is a hierarchical structure consisting of a Poisson point process with a random log-intensity given by a Gaussian random field. In practice, the Bayesian method is often used to infer the intensity function from the observation data, and the resulting Bayesian inference problems have application in fields such as ecology, geology, seismology, and neuroimaging [25]. Sampling the posterior distribution for the LGCP model is computationally challenging largely due to the high dimensionality of such problems. In this example we consider a two-dimensional spatial Bayesian inference problem with the LGCP model. Let Ω be a two dimensional $d \times d$ spatial grid, indexed by $\{(i,j)|i,j = 1,...,d\}$, and let $\boldsymbol{X} = \{x_{i,j}\}_{i,j=1}^{d}$ be a Gaussian process defined on Ω: namely the elements in any nonempty subset of $\boldsymbol{X}$ follows a (multivariate) Gaussian distribution. Moreover, we shall assume that the Gaussian process $\boldsymbol{X}$ is of a constant mean μ and a squared exponential covariance function:

$$\Sigma[(i,j),(i',j')] = \alpha \exp(-\delta(i,i',j,j')/\beta d), \text{ with } \delta(i,i',j,j') = \sqrt{(i-i')^2 + (j-j')^2},$$

where α and β are parameters that will be specified later. Next from $\boldsymbol{X}$, we define the latent intensity process $\Lambda = \{\lambda_{i,j}\}$ with means $\lambda_{ij} = s\exp(x_{ij})$ where s is a scalar parameter. Let

parameters	β_0	β_1	β_2	β_3	β_4	β_5	β_6
MCES	−1.20	−0.73	0.42	−0.41	0.13	−0.36	−0.17
	(0.09)	(0.09)	(0.10)	(0.09)	(0.10)	(0.09)	(0.09)
NUTS	−1.20	−0.73	0.42	−0.41	0.13	−0.36	−0.17
	(0.09)	(0.09)	(0.10)	(0.10)	(0.11)	(0.09)	(0.09)
parameters	β_7	β_8	β_9	β_{10}	β_{11}	β_{12}	
MCES	−0.15	0.01	0.18	−0.11	−0.22	0.12	
	(0.08)	(0.09)	(0.10)	(0.10)	(0.08)	(0.09)	
NUTS	−0.15	0.01	0.18	−0.11	−0.22	0.12	
	(0.08)	(0.09)	(0.09)	(0.10)	(0.08)	(0.09)	
parameters	β_{13}	β_{14}	β_{15}	β_{16}	β_{17}	β_{18}	
MCES	0.03	−0.13	−0.29	0.28	−0.30	0.30	
	(0.09)	(0.09)	(0.12)	(0.08)	(0.10)	(0.12)	
NUTS	0.03	−0.14	−0.29	0.28	−0.30	0.30	
	(0.09)	(0.09)	(0.12)	(0.08)	(0.10)	(0.12)	
parameters	β_{20}	β_{21}	β_{22}	β_{23}	β_{24}	β_{19}	
MCES	0.12	−0.06	−0.09	−0.03	−0.02	0.27	
	(0.14)	(0.14)	(0.09)	(0.13)	(0.12)	(0.11)	
NUTS	0.12	−0.06	−0.09	−0.03	−0.02	0.27	
	(0.14)	(0.14)	(0.09)	(0.13)	(0.12)	(0.11)	

TABLE 3.3

The posterior mean and the standard deviation (in parenthesis) computed by MCES and NUTS for the German Credit example.

$\boldsymbol{Y} = \{y_{ij}|i,j,= 1,...,d\}$ be the data set that are observed where data points $y_{i,j}$ are Poisson distributed with mean $\lambda_{i,j}$:

$$y_{i,j} \sim \text{Poisson}(\cdot, \lambda_{i,j}),$$

and conditionally independent given the latent intensity process $\Lambda = \{\lambda_{i,j}\}$. The goal of the problem is to compute the posterior distribution of $\boldsymbol{X}$ given the data set $\boldsymbol{Y}$.

In the numerical tests, we specify the model parameters as is in Table 3.5, and we emphasize here that we take $d = 32$, resulting in a 1024 dimensional inference problem. Moreover in our tests, the ground truth is randomly generated from the prior and then a synthetic data set is obtained from the generative process for this model; both the ground truth and the generated data set are shown in Fig. 3.4. We then sample the posterior distribution using both NUTS and MCES. As the problem is of very high dimensions, we draw 5.5×10^5 samples from the posterior distribution with either method, with the first 50000 samples are used as the burn-in in both methods. We emphasize here that we use such a large number of samples in this example because of the high dimensionality of the problem. We compare the posterior results of both methods in Figs. 3.4: namely the first row of the figures are from left to right respectively the ground truth of the log-intensity x, that of the intensity λ and the observation data; the second row shows the posterior mean and variance computed by NUTS, and the last one shows those posterior statistics computed by the proposed MCES method. One can see that, both the posterior statistics computed by both methods agree quite well with each other, and the posterior means are reasonably close to the ground truths, which

parameters	β_0	β_1	β_2	β_3	β_4
MCES	0.1314	0.1337	0.1399	0.1315	0.1426
	(0.0528)	(0.0622)	(0.0507)	(0.0569)	(0.0540)
NUTS	0.0554	0.0532	0.0524	0.0520	0.0519
	(0.0049)	(0.0036)	(0.0033)	(0.0037)	(0.0035)
parameters	β_5	β_6	β_7	β_8	β_9
MCES	0.1344	0.1406	0.1405	0.1329	0.1373
	(0.0505)	(0.0539)	(0.0479)	(0.0553)	(0.0529)
NUTS	0.0530	0.0524	0.518	0.0523	0.0521
	(0.0033)	(0.0032)	(0.0034)	(0.0034)	(0.0034)
parameters	β_{10}	β_{11}	β_{12}	β_{13}	β_{14}
MCES	0.1272	0.1327	0.1302	0.1414	0.1341
	(0.0533)	(0.0505)	(0.0580)	(0.0546)	(0.0522)
NUTS	0.0526	0.0524	0.0519	0.0522	0.0522
	(0.0033)	(0.0032)	(0.0034)	(0.0029)	(0.0033)
parameters	β_{15}	β_{16}	β_{17}	β_{18}	β_{19}
MCES	0.1294	0.1327	0.1331	0.1346	0.1387
	(0.0517)	(0.0544)	(0.0534)	(0.0504)	(0.0539)
NUTS	0.0543	0.0525	0.0532	0.0521	0.0517
	(0.0033)	(0.0033)	(0.0034)	(0.0034)	(0.0033)
parameters	β_{20}	β_{21}	β_{22}	β_{23}	β_{24}
MCES	0.1418	0.1392	0.1312	0.1449	0.1402
	(0.0543)	(0.0563)	(0.0576)	(0.0514)	(0.0526)
NUTS	0.0518	0.0522	0.0525	0.0522	0.0523
	(0.0037)	(0.0035)	(0.0031)	(0.0037)	(0.0039)

TABLE 3.4
The means and standard deviations of ESS/L for MCES and NUTS in German Credit Example.

Parameter	α	β	μ	d	s
Value	1.91	1/33	$\log 126 - \alpha/2$	32	$1/d^2$

TABLE 3.5
Model parameters for the LGCP example.

validates the samples drawn by both methods. Next we shall compare the sampling efficiency of the two methods, measured by ESS/L, and we show the comparison results in Figs. 3.5. In the left plot of Figs. 3.5 we show the ratio between the ESS/L value of MCES and that of NUTS at each spatial point of the graph, and we can see that at each spatial point, the ratio is near three, which means that the ESS/L value of MCES is around 3 times as much as that of NUTS at each location. Moreover, the performance ratio is rather stable across locations, ranging between 2.8 to 3. In the right figure of Figs. 3.5, we show the scatter plot of the ESS/L value of all the 1024 dimensions: each dimension is represented as a scatter point with ESS/L of NUTS being the abscissa and that of MCES being the ordinate. First we can see from the plot that all the scattering points are bounded by the two lines $y = 2.8x$ and $y = 3x$, indicating that MCES is at least 2.8 times as efficient as

NUTS in terms of ESS/L. In addition the figure also reveals that the ESS/L results for both methods do not vary significantly across dimensions: for NUTS the 1024 results are between 0.031 and 0.032 and for MCEC they are from 0.090 to 0.093.

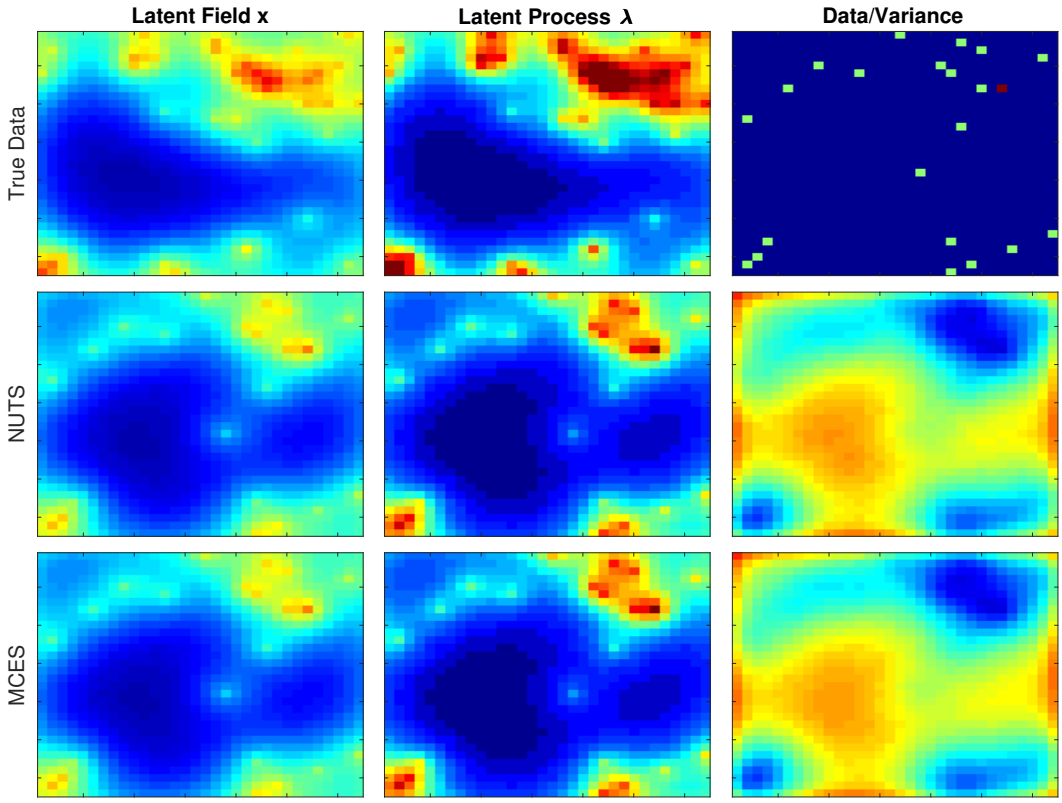

FIG. 3.4. *Comparing quality of posterior distributions from samples obtained using NUTS and MCES for the LGCP model. The top-right image shows the observation data.*

4. Conclusions. In this work we propose a new KSE/CE based design criterion for tuning the algorithm parameters in HMC. We show that the KSE/CE criterion can address some limitation of the distance based design criteria such as ESJD. For near-Gaussian distribution we are able to derive the analytical solution to the resulting optimization problem. We then develop an adaptive HMC algorithm based on the results. Numerical examples demonstrate that the proposed method has rather good performance even when the target distributions are considerably different from Gaussian. Several issues and limitations of the method need to be addressed in the future. First, Algorithm 1 terminates the adaptation after a fixed number of iterations, which may potentially affect the efficiency of the algorithm, if the adaptation is terminated prematurely. To this end, an interesting question is that whether the chain can converge without such a mandatory termination. Moreover, the most serious restriction of the method is, of course, the near-Gaussian assumption, which makes the method unsuitable for strongly non-Gaussian distributions, e.g., those with multiple modes. It is thus of significant interest to apply the KSE/CE criterion to strongly non-Gaussian

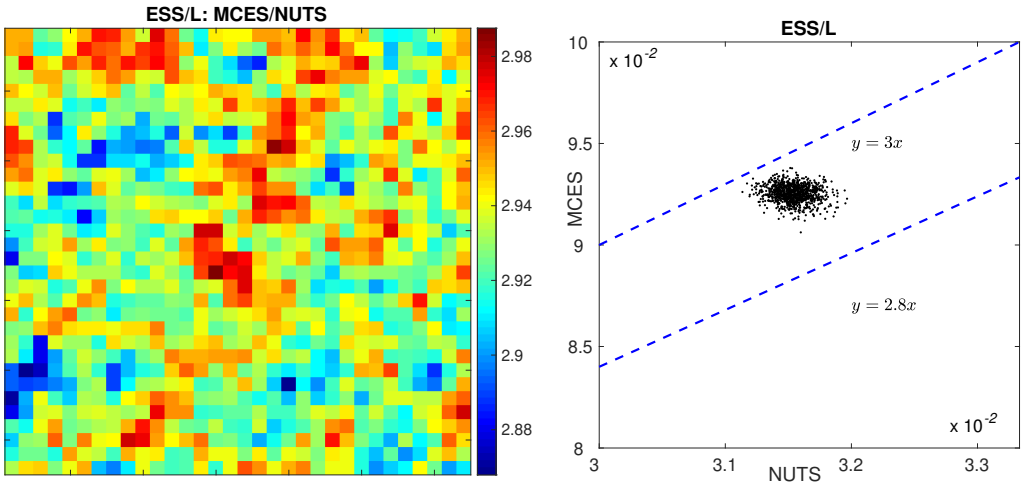

FIG. 3.5. *The ESS/L comparison of MCES and NUTS for the LGCP model. Left: the ratio between the ESS/L value of MCES and that of NUTS at each location. Right: the scatter plot of the ESS/L of the two methods of all the dimensions.*

distributions and develop suitable HMC algorithms for them. Finally, in the present algorithm we adaptively estimate the covariance matrix which in turn defines the kinetic energy of the system, and to this end another interesting extension is to construct the kinetic energy locally as is done in [12], which may be a more feasible solution for strongly non-Gaussian distributions. We plan to investigate these problems in the future.

Appendix A. Derivation of the MCE and the maximum ESJD solutions. This section provides details of the derivation of the optimal integration time with respect to CE and with respect to ESJD when the target distribution is $N(0,k)$. We also take $K(p) = p^2/(2m)$ and $p_0 \sim N(0,m)$. In this case it is easy to derive that the solution of the Hamiltonian system is

$$x(t) = A\cos(\sqrt{\frac{1}{km}}t + \phi_0), \quad p(t) = -A\sqrt{\frac{m}{k}}\sin(\sqrt{\frac{1}{km}}t + \phi_0) \tag{A.1a}$$

and the initial conditions are

$$x_0 = A\cos(\phi_0), \quad p_0 = -A\sqrt{m/k}\sin(\phi_0). \tag{A.1b}$$

From Eq. (A.1), we obtain

$$\begin{aligned} x_T &= A\cos(\sqrt{\frac{1}{km}}T + \phi_0) \\ &= [x_0\cos(\sqrt{\frac{1}{km}}T) + p_0\sqrt{\frac{k}{m}}\sin(\sqrt{\frac{1}{km}}T)]. \end{aligned}$$

As $p_0 \sim N(0, m)$, it follows immediately that

$$\pi(x_T|x_0) = N\left(x_0 \cos(\sqrt{\frac{1}{km}}T), k\sin^2(\sqrt{\frac{1}{km}}T)\right). \tag{A.2}$$

Next we shall consider the two criteria separately.

First we consider the CE criterion, which seeks to maximize $\mathbb{H}[x_T|x_0]$, and since $\pi(x_T|x_0)$ is univariate Gaussian, it is equivalent to

$$\begin{aligned}\max_{T>0} \mathbb{E}_{\mathbf{x}_0}[\log \mathrm{Var}[\mathbf{x}_T|\mathbf{x}_0]] &:= \log[k\sin^2(\sqrt{\frac{1}{km}}T)] \\ &= \log[1-\cos(2\sqrt{\frac{1}{km}}T)] + \log(\frac{k}{2}).\end{aligned} \tag{A.3}$$

It is easy to see that the solution is,

$$T = \frac{1}{2}\sqrt{km}(\pi + 2J\pi),$$

where J is an arbitrary non-negative integer. Certainly we should take $J = 0$ and so we obtain the smallest T as larger T implies higher computational cost of the numerical integration. Thus the optimal solution with respect to the CE criterion is

$$T = \frac{\pi}{2}\sqrt{km}.$$

Next we shall derive the optimal value of T with respect to ESJD. That is we want to solve,

$$\max_{T>0} \mathbb{E}_{x_0,p_0}[||x_T - x_0||^2].$$

Once again from Eq. (A.1) we obtain,

$$\begin{aligned}x_T - x_0 &= A\cos(\sqrt{\frac{1}{km}}T + \phi_0) - A\cos(\phi_0) \\ &= x_0[\cos(\sqrt{\frac{1}{km}}T) - 1] + p_0\sqrt{\frac{k}{m}}\sin(\sqrt{\frac{1}{km}}T).\end{aligned}$$

Then we have,

$$\mathbb{E}_{x_0,p_0}||x_T - x_0||^2 = 2k[1-\cos(\sqrt{\frac{1}{km}}T)].$$

Thus, maximizing the ESJD becomes,

$$\max_{T>0} 2k\left[1-\cos(\sqrt{\frac{1}{km}}T)\right],$$

and the solution is

$$T = \sqrt{km}(\pi + 2J\pi),$$

and for the same reason as above we take $J = 0$, which yields the optimal integration time with respect to the ESJD,

$$T = \pi\sqrt{km}.$$

Appendix B. Additional test results for the Eight School example. We present three additional test results for the Eight School example to demonstrate the robustness of the method against the algorithm parameter values. Specifically we implemented the MCES with another three different sets of parameters shown in the tables B.1-B.3 below. Compared to the parameter values used in Section 3.3, we vary the values of three key parameters: in test 1 we change Acc_{min} from 60% to 40%, in test 2 we change I_{max} from 2 to 1, and in test 3 we change I_{max} from 60 to 100. We plot the ESS/L results for the three tests in Figs. B.1, B.2 and B.3 respectively. The figures demonstrate that, in all the tests, the MCE-HMC method yields evidently better ESS per L results than NUTS, suggesting that the performance of the MCE method is not sensitive to these parameters.

Parameter	L_0	L_{max}	ρ	Acc_{min}	N_{max}	N_L	N_M	N_0	I_{max}
Value	1	60	1.2	40%	10000	200	2000	1000	2

TABLE B.1
Algorithm parameters for test 1.

Parameter	L_0	L_{max}	ρ	Acc_{min}	N_{max}	N_L	N_M	N_0	I_{max}
Value	1	60	1.2	60%	10000	200	2000	1000	1

TABLE B.2
Algorithm parameters for test 2.

Parameter	L_0	L_{max}	ρ	Acc_{min}	N_{max}	N_L	N_M	N_0	I_{max}
Value	1	100	1.2	60%	10000	200	2000	1000	2

TABLE B.3
Algorithm parameters for test 3.

REFERENCES

[1] C. ANDRIEU AND J. THOMS, *A tutorial on adaptive mcmc*, Statistics and computing, 18 (2008), pp. 343–373.

[2] A. BARP, F.-X. BRIOL, A. D. KENNEDY, AND M. GIROLAMI, *Geometry and dynamics for markov chain monte carlo*, Annual Review of Statistics and Its Application, 5 (2018), pp. 451–471.

[3] A. BESKOS, N. PILLAI, G. ROBERTS, J.-M. SANZ-SERNA, A. STUART, ET AL., *Optimal tuning of the hybrid monte carlo algorithm*, Bernoulli, 19 (2013), pp. 1501–1534.

[4] M. BETANCOURT, *A conceptual introduction to hamiltonian monte carlo*, arXiv preprint arXiv: 1701.02434, (2017).

[5] T. BUI-THANH AND M. GIROLAMI, *Solving large-scale pde-constrained bayesian inverse problems with riemann manifold hamiltonian monte carlo*, Inverse Problems, 30 (2014), p. 114014.

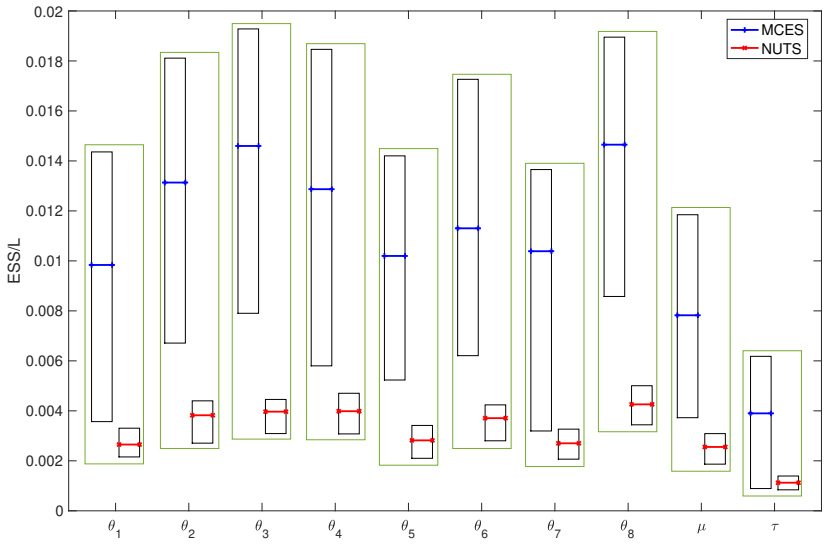

FIG. B.1. *The box plots of the ESS per L for test 1.*

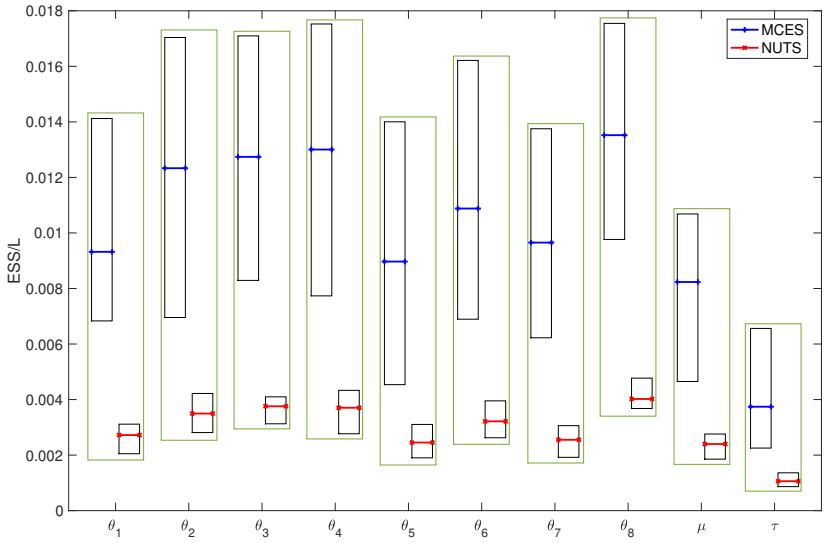

FIG. B.2. *The box plots of the ESS per L for test 2.*

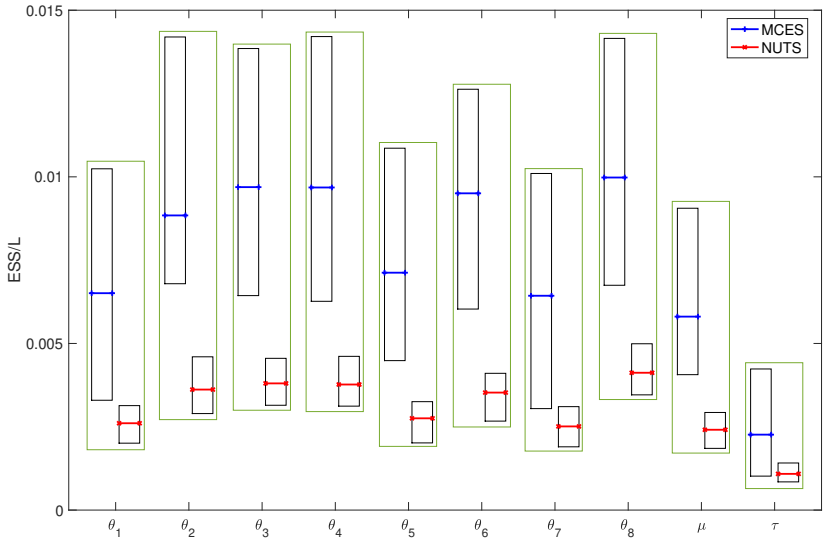

FIG. B.3. *The box plots of the ESS per L for test 3.*

[6] B. CARPENTER, A. GELMAN, M. D. HOFFMAN, D. LEE, B. GOODRICH, M. BETANCOURT, M. BRUBAKER, J. GUO, P. LI, AND A. RIDDELL, *Stan: A probabilistic programming language*, Journal of statistical software, 76 (2017).

[7] T. CHEN, E. FOX, AND C. GUESTRIN, *Stochastic gradient hamiltonian monte carlo*, in International conference on machine learning, 2014, pp. 1683–1691.

[8] L. DEMETRIUS AND T. MANKE, *Robustness and network evolution—an entropic principle*, Physica A: Statistical Mechanics and its Applications, 346 (2005), pp. 682–696.

[9] D. DUA AND C. GRAFF, *UCI machine learning repository*, 2017.

[10] S. DUANE, A. D. KENNEDY, B. J. PENDLETON, AND D. ROWETH, *Hybrid monte carlo*, Physics letters B, 195 (1987), pp. 216–222.

[11] A. GELMAN, H. S. STERN, J. B. CARLIN, D. B. DUNSON, A. VEHTARI, AND D. B. RUBIN, *Bayesian data analysis*, Chapman and Hall/CRC, 2013.

[12] M. GIROLAMI AND B. CALDERHEAD, *Riemann manifold langevin and hamiltonian monte carlo methods*, Journal of the Royal Statistical Society: Series B (Statistical Methodology), 73 (2011), pp. 123–214.

[13] H. HAARIO, M. LAINE, A. MIRA, AND E. SAKSMAN, *Dram: efficient adaptive mcmc*, Statistics and computing, 16 (2006), pp. 339–354.

[14] H. HAARIO, E. SAKSMAN, J. TAMMINEN, ET AL., *An adaptive metropolis algorithm*, Bernoulli, 7 (2001), pp. 223–242.

[15] M. D. HOFFMAN AND A. GELMAN, *The no-u-turn sampler: adaptively setting path lengths in hamiltonian monte carlo.*, Journal of Machine Learning Research, 15 (2014), pp. 1593–1623.

[16] D. W. HOSMER JR, S. LEMESHOW, AND R. X. STURDIVANT, *Applied logistic regression*, vol. 398, John Wiley & Sons, 2013.

[17] T. JAAKKOLA AND M. JORDAN, *A variational approach to bayesian logistic regression models and their extensions*, in Sixth International Workshop on Artificial Intelligence and Statistics, vol. 82, 1997.

[18] M. MIHELICH, B. DUBRULLE, D. PAILLARD, Q. KRAL, AND D. FARANDA, *Maximum kolmogorov-sinai entropy versus minimum mixing time in markov chains*, Journal of Statistical Physics, 170 (2018), pp. 62–68.

[19] J. MOLLER, A. R. SYVERSVEEN, AND R. P. WAAGEPETERSEN, *Log gaussian cox processes*, Scandinavian journal of statistics, 25 (1998), pp. 451–482.

[20] R. M. NEAL, *Mcmc using hamiltonian dynamics*, in Handbook of markov chain monte carlo, S. Brooks, A. Gel-

man, G. Jones, and X.-L. Meng, eds., Chapman & Hall/CRC, 2011, pp. 131–162.
[21] A. Nishimura, *(recycled) no-u-turn-sampler : Matlab implementation*, 2017.
[22] C. Pasarica and A. Gelman, *Adaptively scaling the metropolis algorithm using expected squared jumped distance*, Statistica Sinica, 20 (2007), pp. 343–364.
[23] Y. Sinai, *Kolmogorov-sinai entropy*, Scholarpedia, 4 (2009), p. 2034.
[24] H. Strathmann, D. Sejdinovic, S. Livingstone, Z. Szabo, and A. Gretton, *Gradient-free hamiltonian monte carlo with efficient kernel exponential families*, in Advances in Neural Information Processing Systems, 2015, pp. 955–963.
[25] M. Teng, F. Nathoo, and T. D. Johnson, *Bayesian computation for log-gaussian cox processes: a comparative analysis of methods*, Journal of statistical computation and simulation, 87 (2017), pp. 2227–2252.
[26] Z. Wang, S. Mohamed, and N. Freitas, *Adaptive hamiltonian and riemann manifold monte carlo*, in International Conference on Machine Learning, 2013, pp. 1462–1470.
[27] T. Yu, *Maximum conditional entropy sampler: Matlab implementation.* https://github.com/SiriusYtc/MCES, 2017.
[28] Y. Zhang and C. A. Sutton, *Quasi-newton methods for markov chain monte carlo*, in Advances in Neural Information Processing Systems, 2011, pp. 2393–2401.